Secure Key Encapsulation Mechanism with Compact Ciphertext and Public Key from Generalized Srivastava code

Jayashree Dey and Ratna Dutta

Department of Mathematics, Indian Institute of Technology Kharagpur,
Kharagpur-721302, India
deyjayashree@iitkgp.ac.in,ratna@maths.iitkgp.ernet.in

Abstract. Code-based public key cryptosystems have been found to be an interesting option in the area of Post-Quantum Cryptography. In this work, we present a key encapsulation mechanism (KEM) using a parity check matrix of the Generalized Srivastava code as the public key matrix. Generalized Srivastava codes are privileged with the decoding technique of Alternant codes as they belong to the family of Alternant codes. We exploit the dyadic structure of the parity check matrix to reduce the storage of the public key. Our encapsulation leads to a shorter ciphertext as compared to DAGS proposed by Banegas et al. in Journal of Mathematical Cryptology which also uses Generalized Srivastava code. Our KEM provides IND-CCA security in the random oracle model. Also, our scheme can be shown to achieve post-quantum security in the quantum random oracle model.

Keywords: Key encapsulation mechanism · Generalized Srivastava code · Quasi-dyadic matrix · Alternant decoding.

1 Introduction

Cryptography and coding theory are at the core of implementation of telecommunication systems, computational systems and secure networks. Cryptography based on error correcting codes is one of the main approaches to guarantee secure communication in post-quantum world. The security of current widely used classical cryptosystems relies on the difficulty of number theory problems like factorization and the discrete logarithm problem. P.W. Shor [51] showed in 1994 that most of these cryptosystems can be broken once sufficiently strong quantum computers become available. Thus, it is necessary to devise alternatives that can survive quantum attacks while offering reasonable performance with solid security guarantees. At the end of 2016, the National Institute of Standards and Technology (NIST) announced a call for ideas to develop quantum-resistant cryptographic primitives. Besides, government organizations like the European Commission and the Japanese Society promoted research programs to improve post-quantum cryptography research.

Code-based cryptosystems are usually very fast and can be implemented on several platforms, both software and hardware. They do not require special-purpose hardware, specifically no cryptographic co-processors. The security of code-based cryptography mainly relies on the following two computational assumptions:

(i) the hardness of generic decoding [19] which is NP complete and also believed to be hard on average even against quantum adversaries
(ii) the pseudorandomness of the underlying code $\mathcal{C}$ for the construction which states that it is hard to distinguish a random matrix from a generator (or parity check) matrix of $\mathcal{C}$ used as a part of the public key of the system.

Designing practical alternative cryptosystems based on difficulty of decoding unstructured or random codes is currently a major research area. The public key indistinguishability problem strongly depends on the code family. For instance, the McEliece encryption scheme [39] uses binary Goppa codes for which this indistinguishability assumption holds. On the other hand, the assumption does not hold for other families such as Reed Solomon codes, Concatenated codes, Low Density Parity Check (LDPC) codes etc. In [27], Faugere et al. devise a distinguisher for high rate Goppa codes. One of the key challenge in code-based cryptography is to come up with families of codes for which the indistinguishability assumption holds.

Constructing efficient and secure code-based cryptographic scheme is a challenging task. The crucial fact in designing code-based cryptosystems is to use a linear error-correcting code in such a way that the public key is indistinguishable from a random key. A codeword is used as ciphertext of a carefully chosen linear error-correcting code to which random errors are added. The decryptor with the knowledge of a trapdoor can perform fast polynomial time decoding, remove the errors and recover the plaintext. Attackers are reduced to a generic decoding problem and the system remains secure against an adversary equipped with a quantum computer.

Our Contribution. In this paper, we focus on designing an IND-CCA secure efficient code-based KEM that relies on the difficulty of generic decoding problem. Our starting point is the key encapsulation mechanism DAGS [12] that uses the quasi-dyadic structure of Generalized Srivastava (GS) code. Quasi-dyadic structure reduces the public key size remarkably in DAGS while the encapsulation procedure increases the size of ciphertext. We aim to design a KEM with relatively short ciphertext. We deploy the Niederreiter framework to develop our KEM using a syndrome as ciphertext and achieve IND-CCA security in the random oracle model. More precisely, we use the parity check matrix of the Generalized Srivastava code as the public key and utilize its block dyadic structure to reduce the public key size. We consider the syndrome of a vector as the ciphertext header where the vector is formed by parsing two vectors – the first vector is an error vector that is generated by a deterministic error vector generation algorithm and the second vector is constructed from a hash value of a randomly chosen message by the encapsulator. This significantly reduces the ciphertext header size that makes the scheme useful in application with limited

communication bandwidth. Also, the use of the parity check matrix directly in computing the ciphertext is more fast and efficient. For decapsulation, we form an equivalent parity check matrix using the secret key to decode the ciphertext header and then proceed to get the decapsulation key. Note that, Generalized Srivastava codes belong to the class of Alternant codes which have benefits of an efficient decoding algorithm. The complexity of decoding is $O(n \log^2 n)$ [49] which is the same as that of Goppa codes where n is the length of the code.
Technical Overview. Our $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ employs the quasi-dyadic variant of Generalized Srivastava codes.

In Setup part, the global public parameters $\mathsf{param} = (n, n_0, k, k', w, q, s, t, r, m, \mathcal{G}, \mathcal{H}, \mathcal{H}')$ are generated and published where λ is a security parameter. Here, $q = 2^{p_1},\ s = 2^{p_2}$ and $n = n_0 s$ where n_0, p_1, p_2, m are positive integers and $k = n - mst$. The three functions $\mathcal{G} : (\mathsf{GF}(q))^{k'} \longrightarrow (\mathsf{GF}(q))^{k}$, $\mathcal{H} : (\mathsf{GF}(q))^{k'} \longrightarrow (\mathsf{GF}(q))^{k'}$ and $\mathcal{H}' : \{0,1\}^* \longrightarrow \{0,1\}^r$ are cryptographically secure hash functions where the positive integer r denotes the desired key length.

In key generation, a parity check matrix $\overline{B}$ (over $\mathsf{GF}(q^m)$) of Generalized Srivastava code is constructed and then transformed into a matrix C over base field $\mathsf{GF}(q)$. The matrix C is written in systematic form $(M|I_{n-k})$ where $M = (M_{i,j})$ preserves the dyadic structure. The public key of our KEM is $\mathsf{pk} = \{\boldsymbol{\psi}_{i,j} \mid i = 0, 1, \ldots, mt-1,\ j = 0, 1, \ldots, \frac{k}{s} - 1\}$ where $\boldsymbol{\psi}_{i,j} \in (\mathsf{GF}(q))^s$ is the first row of $M_{i,j}$, $i = 0, 1, \ldots, mt-1$, $j = 0, 1, \ldots, \frac{k}{s} - 1$ and each block matrix $M_{i,j}$ is $s \times s$ dyadic matrix with dyadic signature $\boldsymbol{\psi}_{i,j}$. The secret key is $\mathsf{sk} = (\mathbf{v}, \mathbf{y})$ where $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1}) \in (\mathsf{GF}(q^m))^n$ and $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in (\mathsf{GF}(q^m))^n$ with

$$y_j = \frac{z_j}{\prod\limits_{i=0}^{s-1} (u_i - v_j)^t} \quad \text{for } j = 0, 1, \ldots, n-1.$$

For encapsulation, $\mathbf{r} = \mathcal{G}(\mathbf{m})$, $\mathbf{d} = \mathcal{H}(\mathbf{m})$ are computed after sampling a message $\mathbf{m}$ randomly. The vector $\mathbf{r}$ is parsed as $\mathbf{r} = (\boldsymbol{\rho}||\boldsymbol{\sigma})$ and $\boldsymbol{\mu} = (\boldsymbol{\rho}||\mathbf{m})$. Using the public key $\mathsf{pk} = \{\boldsymbol{\psi}_{i,j} | i = 0, 1, \ldots, mt-1,\ j = 0, 1, \ldots, \frac{k}{s} - 1\}$, the matrix $H = (M|I_{n-k})$ is reconstructed. The parity check matrix H is indistinguishable from a random matrix over $\mathsf{GF}(q)$. The ciphertext header component $\mathbf{c}$ is computed as $\mathbf{c} = H(\mathbf{e}')^T$ where $\mathbf{e}' = (\mathbf{e}||\boldsymbol{\mu})$. The vector $\mathbf{e}$ having length $n - k$ and weight $w - \mathsf{wt}(\boldsymbol{\mu}')$ is generated deterministically using $\boldsymbol{\sigma}$ as a seed. The encapsulator sets the ciphertext header $\mathsf{CT} = (\mathbf{c}, \mathbf{d}) \in (\mathsf{GF}(q))^{n-k+k'}$ and computes encapsulation key $K = \mathcal{H}'(\mathbf{m}) \in \{0,1\}^r$.

In the decapsulation phase, the ciphertext header component $\mathbf{c}$ is decoded to get error $\mathbf{e}''$ using the technique of alternant decoding for Generalized Srivastava code where the parity check matrix H' over $\mathsf{GF}(q^m)$ in alternant form is formed using the secret key $(\mathbf{v}, \mathbf{y})$. The error vector $\mathbf{e}''$ is parsed as $\mathbf{e}'' = (\mathbf{e}_0||\boldsymbol{\mu}')$ with $\boldsymbol{\mu}'$=$(\boldsymbol{\rho}'||\mathbf{m}')$. The vector $\mathbf{r}' = \mathcal{G}(\mathbf{m}')$ is computed extracting $\mathbf{m}'$ from $\boldsymbol{\mu}'$. Let $\mathbf{r}' = (\boldsymbol{\rho}''||\boldsymbol{\sigma}')$ and $\mathbf{d}' = \mathcal{H}(\mathbf{m}')$. After extracting $\boldsymbol{\sigma}'$ from $\mathbf{r}'$, an error vector $\mathbf{e}_0'$ of length $n - k$ and weight $w - \mathsf{wt}(\boldsymbol{\mu}')$ is generated in a deterministic way taking $\boldsymbol{\sigma}'$ as seed. The decapsulator checks whether $(\mathbf{e}_0 \neq \mathbf{e}_0') \vee (\boldsymbol{\rho}' \neq \boldsymbol{\rho}'') \vee (\mathbf{d} \neq \mathbf{d}')$. If the condition holds, the symbol $\perp$ is returned indicating decapsulation failure. Otherwise, the encapsulation key $K = \mathcal{H}'(\mathbf{m}')$ is returned. A parity check matrix in alternant form over $\mathsf{GF}(q^m)$ is required for decoding. The parity check matrix H over $\mathsf{GF}(q)$ derived from the public key pk is a parity check matrix of Generalized Srivastava code and does not provide advantages to decode the ciphertext header component $\mathbf{c}$ as the syndrome decoding problem is hard over the field $\mathsf{GF}(q)$. To apply alternant decoding for Generalized Srivastava code, we need a parity check matrix H' over $\mathsf{GF}(q^m)$ which is derived from the secret key $(\mathbf{v}, \mathbf{y})$.

Table 1. Summary of IND-CCA secure KEMs using random oracles

Scheme	pk size (in bits)	sk size (in bits)	CT size (in bits)	Code used	Cyclic/Dyadic	Correctness error
NTS-KEM [2]	$(n-k)k$	$2(n-k+r)m$ $+nm+r$	$(n-k+r)$	Binary Goppa code	–	No
BIKE-1 [4]	n	$n + w \cdot \lceil \log_2 k \rceil$	n	MDPC code	Quasi-Cyclic	Yes
BIKE-2 [4]	k	$n + w \cdot \lceil \log_2 k \rceil$	k	MDPC code	Quasi-Cyclic	Yes
BIKE-3 [4]	n	$n + w \cdot \lceil \log_2 k \rceil$	n	MDPC code	Quasi-Cyclic	Yes
Classic McEliece [20]	$k(n-k)$	$n+mt+mn$	$(n-k)+r$	Binary Goppa code	–	No
BIG QUAKE [14]	$\frac{k}{\ell}(n-k)$	$mt+mn$	$(n-k)+2r$	Binary Goppa code	Quasi-Cyclic	No
DAGS [12]	$\frac{k}{s}(n-k)\log_2 q$	$2mn\log_2 q$	$[n+k']\log_2 q$	GS code	Quasi-Dyadic	No
This work	$\frac{k}{s}(n-k)\log_2 q$	$2mn\log_2 q$	$[k'+(n-k)]\log_2 q$	GS code	Quasi-Dyadic	No

pk= Public key, sk= Secret key, CT=Ciphertext, k=dimension of the code, n=length of the code, ℓ= length of each blocks, t=error correcting capacity, $k' < k, s, r, w, p_1, p_2$ are positive integers ($\ell << s$), $s = 2^{p_2}$, $q = 2^{p_1}$, m= the degree of field extension, r= the desired key length, GS=Generalized Srivastava, MDPC=Moderate Density Parity Check

In Table 1, we provide a theoretical comparison of our KEM with other recently proposed code-based KEMs. All the schemes in the table are based on finite fields having characteristic 2. We summarize the following features of our KEM.

• The closest related work to ours is DAGS [12]. Similar to DAGS, we also use quasi-dyadic form of Generalized Srivastava code. However, DAGS uses generator matrix whereas we use parity check matrix. Consequently, in our construction, the ciphertext size is reduced by $k \log_2 q$ bits as compared to DAGS [12] whereas the public key and the secret key sizes remain the same. Furthermore our encapsulation is faster than DAGS.

• The public key sizes in our approach are better than NTS-KEM [2], Classic McEliece [20] and BIG QUAKE [14]. Although the BIKE variants are efficient in terms of key sizes and achieve IND-CCA security, they still suffer from small decoding failure rate. The erlier BIKE variants proposed in [3] have a non-negligible decoding failure rate and only attain IND-CPA security.

In the comparison table, we mostly highlight the KEMs which rely on the error correcting codes that belong to the class of Alternant codes except BIKE variants which uses QC-MDPC codes. We exclude the schemes like LEDAkem [7], RLCE-KEM [53], LAKE [5], Ouroboros-R [41], LOCKER [6], QC-MDPC [54], McNie [36] etc. In fact, the schemes LAKE [5], Ouroboros-R [41], LOCKER

[6] uses rank metric codes (LRPC codes) while RLCE-KEM [53] is based on a random linear code and McNie [36] relies on any error correcting code, specially QC-LRPC codes. LEDAkem [7] uses QC-LDPC codes and has a small decoding failure rate. Moreover, it has risks in case of keypair reuse which may cause a reaction attack [26] for some particular instances. The schemes proposed in [1] are also kept out as both HQC and RQC are constructed for any decodable linear code. Also, HQC has a decryption failure and RQC uses rank metric codes. The protocol QC-MDPC may have a high decoding failure rate [50] for some particular parameters which enhances the risk of GJS attack [34]. The KEM protocol CAKE [18] is another important KEM which is merged with another independent construction Ouroboros [24] to obtain BIKE [4].

To prove our KEM's security, we follow the generic transformations in [35]. We construct a public key encryption scheme PKE_1 from our KEM and show that the OW-VA (One-Wayness under Validity Attacks) security of PKE_1 implies the $\mathsf{IND\text{-}CCA}$ security of our KEM considering $\mathcal{H}'$ as a random oracle. Also, $\mathsf{OW\text{-}PCVA}$ (One-Wayness under Plaintext and Validity Checking Attacks) security always implies $\mathsf{OW\text{-}VA}$ security with zero queries to the plaintext checking oracle. We form another public key encryption scheme PKE_2 from PKE_1 and show that breaking $\mathsf{OW\text{-}PCVA}$ security of the PKE_1 would lead to breaking the IND-CPA security of the encryption scheme PKE_2 treating $\mathcal{G}$ as a random oracle. This means $\mathsf{OW\text{-}PCVA}$ security and consequently $\mathsf{OW\text{-}VA}$ security of PKE_1 implies $\mathsf{IND\text{-}CPA}$ security of PKE_2. Finally, we show that PKE_2 achieves $\mathsf{IND\text{-}CPA}$ security under the hardness of syndrome decoding problem and the indistinguishability of the public key matrix. Therefore, we arrive at the following result.

Theorem 1. *(Informal) Assuming the hardness of decisional syndrome decoding problem and indistinguishability of the public key matrix H (derived from the public key pk by running $\mathsf{KEM.KeyGen(param)}$ where $\mathsf{param} \longleftarrow \mathsf{KEM.Setup}(1^\lambda)$, λ being the security parameter), our $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ provides $\mathsf{IND\text{-}CCA}$ security when the hash functions $\mathcal{H}'$ and $\mathcal{G}$ are modeled as random oracles.*

We can extend our security proof in the quantum random oracle following [35] and get the following result.

Theorem 2. *(Informal) Assuming the hardness of decisional syndrome decoding problem and indistinguishability of the public key matrix H (derived from the public key pk by running $\mathsf{KEM.KeyGen(param)}$ where $\mathsf{param} \longleftarrow \mathsf{KEM.Setup}(1^\lambda)$, λ being the security parameter), our $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ provides $\mathsf{IND\text{-}CCA}$ security when the hash functions $\mathcal{G}, \mathcal{H}$ and $\mathcal{H}'$ are modeled as quantum random oracles.*

Related work. The first modern idea to construct a cryptosystem based on coding theory was a public key encryption scheme designed by R. J. McEliece in 1978 [39] which has resisted all cryptanalytic attempts so far. McEliece's original idea was to use a codeword of a binary Goppa code as a ciphertext. An arbitrary basis of the code (i.e. a generator matrix) is the public key that allows an encryptor to encrypt a message. However, it suffers from a rather large public key size. A dual variant of the system, proposed by Niederreiter [45], provides slightly improved efficiency with equivalent security [37] using a parity check matrix as the public key.

Many proposals were already attempted to solve the issue of large key size by replacing binary Goppa codes with codes that permit more compact designs ([42], [23]). Sevaral attempts could not last due to the attacks in ([30], [28], [29]). LDPC codes have been recommended to instantiate McEliece framework ([44], [10], [11], [8]) which does not seem as a secure option due to the cryptanalysis in [9] and [47]. Attempting alternative codes that allow more compact key representations and also preserve security has recently received considerable attention to the current cryptographic community.

In 2013, Misoczki et al. [43] proposed to use Moderate-Density Parity Check (MDPC) codes with their quasi-cyclic (QC) variant as an alternative featuring compact keys. However, it is necessary to attend as MDPC decoding is probabilistic in nature like LDPC codes. Guo, Johansson and Stankovski [34] presented an attack (GJS attack) against the QC-MDPC variant of the McEliece scheme. In [25], Eaton et al. investigated the underlying causes of this attack and how to fix it. One of the earlier key encapsulation mechanism (KEM) constructions is McBits [21] which is framed on the McEliece setup with binary Goppa codes and produces large public keys.

However, following proposals for KEM are submitted to NIST call for standarization of quantum safe cryptography. In 2017, Gaborit et al. suggested BIG QUAKE [14], a Niederreiter type KEM construction where the quasi-cyclic structure of binary Goppa codes is exploited. The protocol chooses very large parameters to provide security against the attack in [29]. In [54], Yamada et al. present QC-MDPC KEM based on McEliece encryption framework with QC-MDPC codes where the quasi-cyclic property leads short keys. The scheme McNie [36] combines the McEliece and Niederreiter public key encryption schemes and uses any error-correcting code having an efficient decoding technique. Employing rank-metric codes, specifically quasi-cyclic Low Rank Parity Check (LRPC) codes can provide compact key sizes. In [1], Aguilar et al. proposed the first efficient code-based cryptosystem whose security depends on decoding vectors having small weights of random quasi-cyclic codes. They provided a reduction of the cryptosystem to this problem together with a thorough analysis of the probability of decryption failure. The authors suggested two new approaches within the structure – the Hamming Quasi-Cyclic (HQC) cryptosystem and the Rank Quasi-Cyclic (RQC) cryptosystem where HQC uses hamming metric and RQC uses rank metric. Besides, the schemes are privileged with a very fast decryption procedure together with short keys. In [20], a KEM, named Classic McEliece, is developed using binary Goppa codes that provides IND-CCA security. In the second round of submission to NIST call, the list of parameter sets has been extended. The scheme NTS-KEM [2] is a variant of the McEliece and Niederreiter encryption schemes. Although neither McEliece nor Niederreiter provides security under either IND-CPA or IND-CCA game individually, NTS-

KEM accomplishes IND-CCA security in the random oracle model by integrating a transform related to the Fujisaki-Okamoto [32] transforms. NTS-KEM utilizes binary Goppa codes and features comparatively compact ciphertexts. Barreto et al. proposed BIKE [3], a suite of algorithms for key encapsulation based on QC-MDPC codes that can be decoded using bit flipping decoding technique with IND-CPA security. The protocol is derived by merging two independent constructions CAKE [18] and Ouroboros [24]. Later, in [4], backflip decoder is used to achieve negligible decoding failure rates and IND-CCA secure BIKE variants. In 2017, Banegas et al. [13] proposed DAGS which is an IND-CCA secure KEM based on Quasi-Dyadic Generalized Srivastava codes. However, an attack had been put in light by the work of [15]. Later, in 2018, Banegas et al. [12] came up with a new construction to reduce key sizes and suggested new sets of parameters to avoid the attack. Recently, another modification of DAGS was proposed with a new framework and a new parameter set in [22]. Though it provides better security, the key sizes (specially the secret key size) are very large.

Other code-based KEM in NIST list are LEDAkem [7] based on QC-LDPC codes, RLCE-KEM [53] based on random linear code, LAKE [5] based on LRPC codes, Ouroboros-R [41] based on quasi-cyclic LRPC codes, LOCKER [6] based on LRPC codes. Very recently, LAKE [5], Ouroboros-R [41] and LOCKER [6] are merged to yield the scheme ROLLO [40].

Organization of the Paper. This rest of the paper is organized as follows. In Section 2, we describe necessary background related to our work. We illustrate our approach to design a KEM in Section 3 and discuss its security in Section 4. Finally, we conclude in Section 5.

2 Preliminaries

In this section, we provide mathematical background and preliminaries that are necessary to follow the discussion in the paper.

Notation. We use the notation $\mathsf{x} \overset{U}{\longleftarrow} X$ for choosing a random element from a set or distribution, $\mathsf{wt}(\mathbf{x})$ to denote the weight of a vector $\mathbf{x}$, $(\mathbf{x}||\mathbf{y})$ for the concatenation of the two vectors $\mathbf{x}$ and $\mathbf{y}$. The matrix I_n is the $n \times n$ identity matrix. We let $\mathbb{Z}^+$ to represent the set $\{a \in \mathbb{Z} | a \geq 0\}$ where $\mathbb{Z}$ is the set of integers. We denote the transpose of a matrix A by A^T and concatenation of the two matrices A and B by $[A|B]$.

2.1 Public Key Encryption

Definition 1. (Public Key Encryption) A public key encryption (PKE) scheme is a tuple PKE=(Setup, KeyGen, Enc, Dec) of four probabilistic polynomial time algorithms (PPT) with the following specifications.

- PKE.Setup$(1^\lambda) \longrightarrow$ param : A trusted authority runs the Setup algorithm which takes a security parameter λ as input and publishes the global public parameters param.

- $\mathsf{PKE.KeyGen}(\mathsf{param}) \longrightarrow (\mathsf{pk}, \mathsf{sk})$: The key generation algorithm, run by a user, takes param as input and returns a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$. The public key pk is published while the secret key sk is kept secret to the user.
- $\mathsf{PKE.Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathsf{CT}$: The encryption algorithm, run by an encryptor, outputs a ciphertext $\mathsf{CT} \in \mathcal{C}$ using a randomness $\mathbf{r} \in \mathcal{R}$ on input the public key pk, a plaintext $\mathbf{m} \in \mathcal{M}$ and the public parameters param. Here $\mathcal{M}$ is the message space, $\mathcal{C}$ is the ciphertext space and $\mathcal{R}$ is the space of randomness.
- $\mathsf{PKE.Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT}) \longrightarrow \mathbf{m} \vee \perp$: A decryptor runs the decryption algorithm that takes the secret key sk, a ciphertext $\mathsf{CT} \in \mathcal{C}$ and public parameters param as input and gets either a plaintext $\mathbf{m} \in \mathcal{M}$ or $\perp$ where the symbol $\perp$ indicates the decryption failure.

Correctness. A PKE scheme is δ-correct if for any security parameter λ, $\mathsf{param} \longleftarrow \mathsf{PKE.Setup}(1^\lambda)$, $(\mathsf{pk},\ \mathsf{sk}) \longleftarrow \mathsf{PKE.KeyGen}(\mathsf{param})$ and $\mathsf{CT} \longleftarrow \mathsf{PKE.Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r})$, it holds that $\Pr[\mathsf{PKE.Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT}) \neq \mathbf{m}] \leq \delta$. The PKE scheme is said to be correct if $\delta = 0$.

Definition 2. (γ-uniformity of PKE [31]). For $\mathbf{m} \in \mathcal{M}$, $\mathsf{param} \longleftarrow \mathsf{PKE.Setup}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{PKE.KeyGen}(\mathsf{param})$, a PKE scheme is said to be γ-uniform if for every possible ciphertext $\mathsf{CT} \in \mathcal{C}$,

$$\Pr_{r \longleftarrow \mathcal{R}}[\mathsf{CT} \longleftarrow \mathsf{PKE.Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r})] \leq \gamma$$

for a real number γ.

A PKE scheme is said to be γ-spread if it is $2^{-\gamma}$-uniform.

Definition 3. (Indistinguishability under Chosen Plaintext Attack (IND-CPA) [33]). The IND-CPA game between a challenger $\mathcal{S}$ and a PPT adversary $\mathcal{A}$ for a public key encryption scheme PKE=(Setup, KeyGen, Enc, Dec) is described below.

1. The challenger $\mathcal{S}$ generates $\mathsf{param} \longleftarrow \mathsf{PKE.Setup}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{PKE.KeyGen}(\mathsf{param})$ where λ is a security parameter and sends $\mathsf{param}, \mathsf{pk}$ to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ sends a pair of messages $\mathbf{m}_0, \mathbf{m}_1 \in \mathcal{M}$ of the same length to $\mathcal{S}$.
3. The challenger $\mathcal{S}$ picks a random bit $b \in \{0, 1\}$, computes a challenge ciphertext $\mathsf{CT} \longleftarrow \mathsf{PKE.Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}_b; \mathbf{r}_b)$ and sends it to $\mathcal{A}$.
4. The adversary outputs a bit b'.

The adversary $\mathcal{A}$ wins the game if $b' = b$. We define the advantage of $\mathcal{A}$ against the above $\mathsf{IND\text{-}CPA}$ security game for the PKE scheme as

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) = |\Pr[b' = b] - 1/2|.$$

A PKE scheme is $\mathsf{IND\text{-}CPA}$ secure if $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) < \epsilon$ where $\epsilon > 0$ is arbitrarily small.
We also define the following four security notions for PKE scheme that are

(i) One-Wayness under Chosen Plaintext Attacks ($\mathsf{OW\text{-}CPA}$), (ii) One-Wayness under Plaintext Checking Attacks ($\mathsf{OW\text{-}PCA}$), (iii) One-Wayness under Validity Checking Attacks ($\mathsf{OW\text{-}VA}$) and (iv) One-Wayness under Plaintext and Validity Checking Attacks ($\mathsf{OW\text{-}PCVA}$).

Definition 4. ($\mathsf{OW\text{-}ATK}$ [35]). For $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{PCA}, \mathsf{VA}, \mathsf{PCVA}\}$, the $\mathsf{OW\text{-}ATK}$ game between a challenger $\mathcal{S}$ and a PPT adversary $\mathcal{A}$ for a public key encryption scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is outlined below where $\mathcal{A}$ can make polynomially many queries to the oracle O_{ATK} given by

$$O_{\mathsf{ATK}} = \begin{cases} - & \mathsf{ATK} = \mathsf{CPA} \\ \mathsf{PCO}(\cdot,\cdot) & \mathsf{ATK} = \mathsf{PCA} \\ \mathsf{CVO}(\cdot) & \mathsf{ATK} = \mathsf{VA} \\ \mathsf{PCO}(\cdot,\cdot), \mathsf{CVO}(\cdot) & \mathsf{ATK} = \mathsf{PCVA} \end{cases}$$

with the Plaintext Checking Oracle $\mathsf{PCO}(\cdot,\cdot)$ and Ciphertext Validity Oracle $\mathsf{CVO}(\cdot)$ as described in Figure 1.

1. The challenger $\mathcal{S}$ generates $\mathsf{param} \longleftarrow \mathsf{PKE.Setup}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{PKE.KeyGen}(\mathsf{param})$ where λ is a security parameter and sends $\mathsf{param}, \mathsf{pk}$ to $\mathcal{A}$.
2. The challenger $\mathcal{S}$ chooses a message $\mathbf{m}^* \in \mathcal{M}$, computes the challenge ciphertext $\mathsf{CT}^* \longleftarrow \mathsf{PKE.Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}^*; \mathbf{r}^*)$ and sends it to $\mathcal{A}$.
3. The adversary $\mathcal{A}$ having access to the oracle O_{ATK}, outputs $\mathbf{m}'$.

The adversary $\mathcal{A}$ wins the game if $\mathbf{m}' = \mathbf{m}^*$. We define the advantage of $\mathcal{A}$ against the above $\mathsf{OW\text{-}ATK}$ security game for PKE scheme as $\mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{OW\text{-}ATK}}(\mathcal{A}) = \Pr[\mathbf{m}' = \mathbf{m}^*]$. The PKE scheme is said to be $\mathsf{OW\text{-}ATK}$ secure if $\mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{OW\text{-}ATK}}(\mathcal{A}) < \epsilon$ for arbitrarily small non zero ϵ.

$\mathsf{PCO}(\mathbf{m} \in \mathcal{M}, \mathsf{CT})$

```
1. if $\mathsf{PKE.Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT}) \longrightarrow \mathbf{m}$
2.     return 1;
3. else
4.     return 0;
5. end if
```

$\mathsf{CVO}(\mathsf{CT} \neq \mathsf{CT}^*)$

```
1. $\mathbf{m} \longleftarrow \mathsf{PKE.Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT})$;
2. if $\mathbf{m} \in \mathcal{M}$
3.     return 1;
4. else
5.     return 0;
6. end if
```

Fig. 1. Plaintext Checking Oracle $\mathsf{PCO}(\cdot,\cdot)$ and Ciphertext Validity Oracle $\mathsf{CVO}(\cdot)$ for $\mathsf{OW\text{-}ATK}$ security game, $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{PCA}, \mathsf{VA}, \mathsf{PCVA}\}$

Remark 1. [35] For any adversary $\mathcal{B}$ there exists an adversary $\mathcal{A}$ with the same running time as that of $\mathcal{B}$ such that $\mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{OW\text{-}CPA}}(\mathcal{B}) \leq \mathsf{Adv}_{\mathsf{PKE}}^{\mathsf{IND\text{-}CPA}}(\mathcal{A}) + 1/|\mathcal{M}|$ where $\mathcal{M}$ is the message space.

Remark 2. An $\mathsf{OW\text{-}PCVA}$ security is also an $\mathsf{OW\text{-}VA}$ security with zero queries to the $\mathsf{PCO}(\cdot,\cdot)$ oracle.

2.2 Key Encapsulation Mechanism

Definition 5. (Key Encapsulation Mechanism). A key encapsulation mechanism (KEM) is a tuple of four PPT algorithms $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ with the following requirements.

- $\mathsf{KEM.Setup}(1^\lambda) \longrightarrow \mathsf{param}$: A trusted authority runs the Setup algorithm which takes a security parameter λ as input and publishes the global public parameters param.
- $\mathsf{KEM.KeyGen}(\mathsf{param}) \longrightarrow (\mathsf{pk}, \mathsf{sk})$: The key generation algorithm, run by a user, takes public parameters param as input and outputs a public-secret key pair $(\mathsf{pk}, \mathsf{sk})$. The public key pk is published while the secret key sk is kept secret to the user.
- $\mathsf{KEM.Encaps}(\mathsf{param}, \mathsf{pk}) \longrightarrow (\mathsf{CT}, K)$: An encapsulator runs the encapsulation algorithm that takes the public key pk and public parameters param as input and outputs a ciphertext header $\mathsf{CT} \in \mathcal{C}$ together with a key $K \in \mathcal{K}$. The ciphertext header CT is broadcasted publicly and the encapsulation key K is kept secret to the encapsulator. Here $\mathcal{C}$ is the ciphertext space and $\mathcal{K}$ is the key space.
- $\mathsf{KEM.Decaps}(\mathsf{param}, \mathsf{sk}, \mathsf{CT}) \longrightarrow K \vee \perp$: A decapsulator runs the decapsulation algorithm on inputs the secret key sk, a ciphertext header CT and public parameters param. It returns the key K or $\perp$ where $\perp$ is a designated symbol indicating failure.

Correctness. A KEM is δ-correct if for any security parameter λ, $\mathsf{param} \longleftarrow \mathsf{KEM.Setup}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{KEM.KeyGen}(\mathsf{param})$ and $(\mathsf{CT}, K) \longleftarrow \mathsf{KEM.Encaps}$ $(\mathsf{param}, \mathsf{pk})$, it holds that $\Pr[\mathsf{KEM.Decaps}(\mathsf{param}, \mathsf{sk}, \mathsf{CT}) \neq K] \leq \delta$. The KEM is correct if $\delta = 0$.

Definition 6. (Indistinguishability under Chosen Ciphertext Attack (IND-CCA) [48]). The $\mathsf{IND\text{-}CCA}$ game between a challenger $\mathcal{S}$ and a PPT adversary $\mathcal{A}$ for a key encapsulation mechanism KEM=(Setup, KeyGen, Encaps, Decaps) is described below.

1. The challenger $\mathcal{S}$ generates $\mathsf{param} \longleftarrow \mathsf{KEM.Setup}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{KEM.KeyGen}(\mathsf{param})$ where λ is a security parameter and sends $\mathsf{param}, \mathsf{pk}$ to $\mathcal{A}$.
2. The PPT adversary $\mathcal{A}$ has access to the decapsulation oracle $\mathsf{KEM.Decaps}$ to which $\mathcal{A}$ can make polynomially many ciphertext queries CT_i and gets the corresponding key $K_i \in \mathcal{K}$ from $\mathcal{S}$.
3. The challenger $\mathcal{S}$ picks a random bit b from $\{0, 1\}$, runs $\mathsf{KEM.Encaps}(\mathsf{param}, \mathsf{pk})$ to generate a ciphertext-key pair (CT^*, K_0^*) with $\mathsf{CT}^* \neq \mathsf{CT}_i$, selects randomly $K_1^* \in \mathcal{K}$ and sends the pair (CT^*, K_b^*) to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ having the pair (CT^*, K_b^*) keeps performing polynomially many decapsulation queries on $\mathsf{CT}_i \neq \mathsf{CT}^*$ and outputs b'.

The adversary succeeds the game if $b' = b$. We define the advantage of $\mathcal{A}$ against the above $\mathsf{IND\text{-}CCA}$ security game for the KEM as

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA}}(\mathcal{A}) = |\Pr[b' = b] - 1/2|.$$

A KEM is $\mathsf{IND\text{-}CCA}$ secure if $\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{A}) < \epsilon$ where $\epsilon > 0$ is arbitrarily small.

2.3 Hardness Assumptions

Definition 7. ((search) (q-ary) Syndrome Decoding (SD) Problem [16]). Given a full-rank matrix $H_{(n-k)\times n}$ over $\mathsf{GF}(q)$, a vector $\mathbf{c} \in (\mathsf{GF}(q))^{n-k}$ and a non-negative integer w, find a vector $\mathbf{e} \in (\mathsf{GF}(q))^n$ of weight w such that $H\mathbf{e}^T = \mathbf{c}$.

More formally, suppose $\mathcal{D}$ is a probabilistic polynomial time algorithm and $U_{(n-k)\times n}$ be the uniform distribution over $(n-k)\times n$ random q-ary matrices. For every positive integer λ, we define the advantage of $\mathcal{D}$ in solving the SD problem by

$$\mathsf{Adv}^{\mathsf{OW}}_{\mathcal{D},\mathsf{SD}}(\lambda) = \Pr[\mathcal{D}(H, H\mathbf{e}^T) = \mathbf{e} | H \xleftarrow{U} U_{(n-k)\times n}, \mathbf{e} \in (\mathsf{GF}(q))^n].$$

Also, we define $\mathsf{Adv}^{\mathsf{OW}}_{\mathsf{SD}}(\lambda) = \max_{\mathcal{D}}[\mathsf{Adv}^{\mathsf{OW}}_{\mathcal{D},\mathsf{SD}}(\lambda)]$ where the maximum is taken over all $\mathcal{D}$. The SD problem is said to be hard if $\mathsf{Adv}^{\mathsf{OW}}_{\mathsf{SD}}(\lambda) < \delta$ where $\delta > 0$ is arbitrarily small.
The corresponding decision problem is proven to be NP-complete [19] in case of binary codes. Later, A. Barg proved that this result holds for codes over all finite fields [16].

Definition 8. ((Decision) (q-ary) Syndrome Decoding (SD) Problem [16]). Given a full-rank matrix $H_{(n-k)\times n}$ over $\mathsf{GF}(q)$, a vector $\mathbf{e} \in (\mathsf{GF}(q))^n$ and a non-negative integer w, is it possible to distinguish between a random syndrome $\mathbf{s}$ and the syndrome $H\mathbf{e}^T$ associated to a w-weight vector $\mathbf{e}$?

Suppose $\mathcal{D}$ is a probabilistic polynomial time algorithm and $U_{(n-k)\times n}$ be the uniform distribution over $(n-k)\times n$ random q-ary matrices. For every positive integer λ, we define the advantage of $\mathcal{D}$ in solving the decisional SD problem by

$$\begin{aligned}\mathsf{Adv}^{\mathsf{DEC}}_{\mathcal{D},\mathsf{SD}}(\lambda) = |&\Pr[\mathcal{D}(H, H\mathbf{e}^T) = 1 \mid \mathbf{e} \in (\mathsf{GF}(q))^n, H \xleftarrow{U} U_{(n-k)\times n}] \\ &- \Pr[\mathcal{D}(H, \mathbf{s}) = 1 \mid \mathbf{s} \xleftarrow{U} U_{(n-k)\times 1}, H \xleftarrow{U} U_{(n-k)\times n}]|\end{aligned}$$

Also, we define $\mathsf{Adv}^{\mathsf{DEC}}_{\mathsf{SD}}(\lambda) = \max_{\mathcal{D}}[\mathsf{Adv}^{\mathsf{DEC}}_{\mathcal{D},\mathsf{SD}}(\lambda)]$ where the maximum is taken over all $\mathcal{D}$. The decisional SD problem is said to be hard if $\mathsf{Adv}^{\mathsf{DEC}}_{\mathsf{SD}}(\lambda) < \delta$ where $\delta > 0$ is arbitrarily small.
In addition, some code based schemes require the following computational assumption. Most of the schemes output a public key that is either a generator matrix or a parity check matrix by running key generation algorithm.

Definition 9. (Indistinguishability of public key matrix H [46]). Let $\mathcal{D}$ be a probabilistic polynomial time algorithm and $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme that uses an $(n-k)\times n$ matrix H as a public key over $\mathsf{GF}(q)$. For every positive integer λ, we define the advantage of $\mathcal{D}$ in distinguishing the public key matrix H from a random matrix R as

$\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{D},H}(\lambda) = \Pr[\mathcal{D}(H) = 1 | (\mathsf{pk} = H, \mathsf{sk}) \longleftarrow \mathsf{PKE.KeyGen}(\mathsf{param}), \mathsf{param} \longleftarrow \mathsf{PKE.Setup}(1^\lambda)] - \Pr[\mathcal{D}(R) = 1 | R \overset{U}{\longleftarrow} U_{(n-k)\times n}]$
where $U_{(n-k)\times n}$ is the uniform distribution over $(n-k)\times n$ random q-ary matrices. We define $\mathsf{Adv}^{\mathsf{IND}}_{H}(\lambda) = \max\limits_{\mathcal{D}}[\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{D},H}(\lambda)]$ where the maximum is over all $\mathcal{D}$. The matrix H is said to be indistinguishable if $\mathsf{Adv}^{\mathsf{IND}}_{H}(\lambda) < \delta$ where $\delta > 0$ is arbitrarily small.

2.4 Basic Definitions from Coding Theory

Definition 10. (Generalized Reed-Solomon Code [38]). Let q be the prime power and m, n be two positive integers. Consider a vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ where α_i*'s* are distinct elements of $\mathsf{GF}(q^m)$. Let $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ be another vector where v_i*'s* are nonzero but not necessarily distinct elements of $\mathsf{GF}(q^m)$. The generalized Reed-Solomon code $\mathsf{GRS}_k(\boldsymbol{\alpha}, \mathbf{v})$ of dimension k associated to $(\boldsymbol{\alpha}, \mathbf{v})$ is a linear code and consists of all vectors

$$\{v_1P(\alpha_1), v_2P(\alpha_2), \ldots, v_nP(\alpha_n)\}$$

where $P(z)$ is any polynomial of degree less than k with coefficients from $\mathsf{GF}(q^m)$. The vectors $\boldsymbol{\alpha}$ and $\mathbf{v}$ are called respectively the support and the multiplier of the code. A parity check matrix of the code is an $r \times n$ matrix of the form

$$H = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \\ \alpha_1 y_1 & \alpha_2 y_2 & \cdots & \alpha_n y_n \\ \alpha_1^2 y_1 & \alpha_2^2 y_2 & \cdots & \alpha_n^2 y_n \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_1^{r-1} y_1 & \alpha_2^{r-1} y_2 & \cdots & \alpha_n^{r-1} y_n \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \cdots & \alpha_n^{r-1} \end{bmatrix} \begin{bmatrix} y_1 & 0 & 0 & \cdots & 0 \\ 0 & y_2 & 0 & \cdots & 0 \\ 0 & 0 & y_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & y_n \end{bmatrix}$$

where $r = n - k$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ is a vector with $y_i \in \mathsf{GF}(q^m),\ y_i \neq 0$ such that $\mathsf{GRS}_k(\boldsymbol{\alpha}, \mathbf{v})^\perp = \mathsf{GRS}_r(\boldsymbol{\alpha}, \mathbf{y})$. Here $\mathsf{GRS}_k(\boldsymbol{\alpha}, \mathbf{v})^\perp$ denotes the dual of $\mathsf{GRS}_k(\boldsymbol{\alpha}, \mathbf{v})$, n is the length of the code and k is the dimension of the code.

Definition 11. (Alternant code [38]). The Alternant code $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$ is the restriction of $\mathsf{GRS}_k(\boldsymbol{\alpha}, \mathbf{v})$ to $\mathsf{GF}(q)$ and consists of all codewords of $\mathsf{GRS}_k(\boldsymbol{\alpha}, \mathbf{v})$ having components in $\mathsf{GF}(q)$.
An alternant code is a linear code defined by the $r \times n$ parity check matrix

$$H = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \\ \alpha_1 y_1 & \alpha_2 y_2 & \cdots & \alpha_n y_n \\ \alpha_1^2 y_1 & \alpha_2^2 y_2 & \cdots & \alpha_n^2 y_n \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_1^{r-1} y_1 & \alpha_2^{r-1} y_2 & \cdots & \alpha_n^{r-1} y_n \end{bmatrix}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are distinct elements of $\mathsf{GF}(q^m)$ and $y_1, y_2, \ldots, y_n$ are non zero elements of $\mathsf{GF}(q^m)$. We refer this particular form of H to be a parity check matrix in alternant form. This code has length n, dimension $k \geq n - mr$, minimum distance $d \geq r+1$, r even and can correct $t = \left\lfloor \frac{d-1}{2} \right\rfloor \leq \frac{r}{2}$ errors. Let $C = (c_{ij}), c_{ij} \in \mathsf{GF}(q^m)$ be any invertible matrix. Then an equivalent parity check matrix for $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$ is

$$H' = CXY = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1r} \\ c_{21} & c_{22} & \cdots & c_{2r} \\ c_{31} & c_{32} & \cdots & c_{3r} \\ \cdots & \cdots & \cdots & \cdots \\ c_{r1} & c_{r2} & \cdots & c_{rr} \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \cdots & \alpha_n^{r-1} \end{bmatrix} \begin{bmatrix} y_1 & 0 & 0 & \cdots & 0 \\ 0 & y_2 & 0 & \cdots & 0 \\ 0 & 0 & y_3 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & y_n \end{bmatrix}$$

$$= \begin{bmatrix} y_1 g_1(\alpha_1) & y_2 g_1(\alpha_2) & \cdots & y_n g_1(\alpha_n) \\ y_1 g_2(\alpha_1) & y_2 g_2(\alpha_2) & \cdots & y_n g_2(\alpha_n) \\ y_1 g_3(\alpha_1) & y_2 g_3(\alpha_2) & \cdots & y_n g_3(\alpha_n) \\ \cdots & \cdots & \cdots & \cdots \\ y_1 g_r(\alpha_1) & y_2 g_r(\alpha_2) & \cdots & y_n g_r(\alpha_n) \end{bmatrix}$$

where $g_i(x) = c_{i1} + c_{i2}x + \cdots + c_{ir}x^{r-1},\ i = 1, 2, \ldots, r$ is a polynomial of degree $< r$ over $\mathsf{GF}(q^m)$.

Definition 12. (Dyadic Matrix and Quasi-dyadic Matrix [17]). Given a ring $\mathbf{R}$ and a vector $\mathbf{h} = (h_0, h_1, \ldots, h_{n-1}) \in \mathbf{R}^n$, the *dyadic* matrix $\Delta(\mathbf{h}) \in \mathbf{R}^{n \times n}$ is a symmetric matrix with components $\Delta_{ij} = h_{i \oplus j}$ where $\oplus$ stands for bitwise exclusive-or. The vector $\mathbf{h}$ is called a signature of the dyadic matrix. The signature of a dyadic matrix forms its first row. A matrix is called *quasi-dyadic* if it is a block matrix whose component blocks are $s \times s$ dyadic submatrices. An $s \times s$ dyadic matrix block can be generated from its first row.

Example : if n is power of 2, then $M = \begin{bmatrix} A & B \\ B & A \end{bmatrix}$ is a $2^k \times 2^k$ dyadic matrix where each block A, B is $2^{k-1} \times 2^{k-1}$ dyadic matrix. Note that, any 1×1 matrix is dyadic.
Generating the dyadic signature [17] : A valid dyadic signature $\mathbf{h} = (h_0, h_1, \ldots, h_{n-1})$ over $\mathbf{R} = \mathsf{GF}(q^m)$ is derived using Algorithm 1 that executes the following two steps.

- Assign nonzero distinct values randomly from $\mathsf{GF}(q^m)$ to h_0 and to every h_i for i a power of 2 and form $\mathbf{h}$ using $\frac{1}{h_{i \oplus j}} = \frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0}$ for appropriate choices of i and j. The resulting signature will have length q^m.
- Return blocks of size s upto length n, making sure to exclude any block containing an undefined entry.

Definition 13. (The Generalized Srivastava (GS) Code [38]). Let $m, n, s, t \in \mathbb{N}$ and q be a prime power. Let $\alpha_1, \alpha_2, \ldots, \alpha_n,\ w_1, w_2, \ldots, w_s$ be $n + s$ distinct elements of $\mathsf{GF}(q^m)$ and $z_1, z_2, \ldots, z_n$ be nonzero elements of $\mathsf{GF}(q^m)$. The Generalized Srivastava (GS) code of length n is a linear code with $st \times n$ parity-check matrix of the form

$$H = \begin{bmatrix} H_1 & H_2 & \cdots & H_s \end{bmatrix}^T$$

Algorithm 1 Constructing a dyadic signature

```
Input: $q$, $m$, $s$, $n$.
Output: A dyadic signature $\mathbf{h} = (h_0, h_1, \dots, h_{n-1})$ over $\mathsf{GF}(q^m)$.
1: repeat
2:     $X = \mathsf{GF}(q^m) \setminus \{0\}$;
3:     $\widehat{h}_0 \xleftarrow{U} X$; $X = X \setminus \{\widehat{h}_0\}$;
4:     for ($l = 0$ to $\lfloor \log q^m \rfloor$) do
5:         $i = 2^l$;
6:         $\widehat{h}_i \xleftarrow{U} X$; $X = X \setminus \{\widehat{h}_i\}$;
7:         for ($j = 1$ to $i - 1$) do
8:             if ($\widehat{h}_i \neq 0 \wedge \widehat{h}_j \neq 0 \wedge \frac{1}{\widehat{h}_i} + \frac{1}{\widehat{h}_j} + \frac{1}{\widehat{h}_0} \neq 0$) then
9:                 $\widehat{h}_{i+j} = \frac{1}{\frac{1}{\widehat{h}_i} + \frac{1}{\widehat{h}_j} + \frac{1}{\widehat{h}_0}}$;
10:            else
11:                $\widehat{h}_{i+j} = 0$; // undefined entry
12:            end if
13:            $X = X \setminus \{\widehat{h}_{i+j}\}$;
14:        end for
15:    end for
16:    $c = 0$;
17:    if ($0 \notin \{\widehat{h}_0, \widehat{h}_1, \dots, \widehat{h}_{s-1}\}$) then
18:        $b_0 = 0; c = 1$; $B_0 = \{\widehat{h}_0, \widehat{h}_1, \dots, \widehat{h}_{s-1}\}$;
19:        for ($j = 1$ to $\lfloor q^m/s \rfloor - 1$) do
20:            if ($0 \notin \{\widehat{h}_{js}, \widehat{h}_{js+1}, \dots, \widehat{h}_{(j+1)s-1}\}$) then
21:                $b_c = j$; $c = c + 1$; $B_c = \{\widehat{h}_{js}, \widehat{h}_{js+1}, \dots, \widehat{h}_{(j+1)s-1}\}$;
22:            end if
23:        end for
24:    end if
25: until ($cs \geq n$)
26: return $\mathbf{h} = (h_0, h_1, \dots, h_{n-1}) = (B_0, B_1, \dots, B_{c-1})$
```

where

$$H_i = \begin{bmatrix} \frac{z_1}{\alpha_1 - w_i} & \frac{z_2}{\alpha_2 - w_i} & \cdots & \frac{z_n}{\alpha_n - w_i} \\ \frac{z_1}{(\alpha_1 - w_i)^2} & \frac{z_2}{(\alpha_2 - w_i)^2} & \cdots & \frac{z_n}{(\alpha_n - w_i)^2} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{z_1}{(\alpha_1 - w_i)^t} & \frac{z_2}{(\alpha_2 - w_i)^t} & \cdots & \frac{z_n}{(\alpha_n - w_i)^t} \end{bmatrix}$$

is a $t \times n$ matrix block. The code is of length $n \leq q^m - s$, dimension $k \geq n - mst$ and minimum distance $d \geq st + 1$. It can correct at most $w = \left\lfloor \frac{d-1}{2} \right\rfloor = \frac{st}{2}$ errors and is an Alternant code. In the parity check matrix

$$H = \begin{bmatrix} y_1 g_1(\alpha_1) & y_2 g_1(\alpha_2) & \cdots & y_n g_1(\alpha_n) \\ y_1 g_2(\alpha_1) & y_2 g_2(\alpha_2) & \cdots & y_n g_2(\alpha_n) \\ y_1 g_3(\alpha_1) & y_2 g_3(\alpha_2) & \cdots & y_n g_3(\alpha_n) \\ \cdots & \cdots & \cdots & \cdots \\ y_1 g_r(\alpha_1) & y_2 g_r(\alpha_2) & \cdots & y_n g_r(\alpha_n) \end{bmatrix}$$

where $g_i(x) = c_{i1} + c_{i2}x + \cdots + c_{ir}x^{r-1},\ i = 1, 2, \ldots, r$ is a polynomial of degree $< r$ over $\mathsf{GF}(q^m)$ for the Alternant code $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$, suppose $r = st$. Also set

$$g_{(l-1)t+k}(x) = \frac{\prod\limits_{j=1}^{s}(x - w_j)^t}{(x - w_l)^k}, l = 1, 2, \ldots, s \text{ and } k = 1, 2, \ldots, t$$

and

$$y_i = \frac{z_i}{\prod\limits_{j=1}^{s}(\alpha_i - w_j)^t}, i = 1, 2, \ldots, n$$

so that

$$y_i g_{(l-1)t+k}(\alpha_i) = \frac{z_i}{(\alpha_i - w_l)^k}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_n,\ w_1, w_2, \ldots, w_s$ be $n + s$ distinct elements of $\mathsf{GF}(q^m)$ and $z_1, z_2, \ldots, z_n$ be nonzero elements of $\mathsf{GF}(q^m)$. The resulting code is a Generalized Srivastava code.

2.5 Decoding of Alternant Code [38]

Suppose $H = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \\ \alpha_1 y_1 & \alpha_2 y_2 & \cdots & \alpha_n y_n \\ \alpha_1^2 y_1 & \alpha_2^2 y_2 & \cdots & \alpha_n^2 y_n \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_1^{r-1} y_1 & \alpha_2^{r-1} y_2 & \cdots & \alpha_n^{r-1} y_n \end{bmatrix}$ be a parity check matrix of an Alternant code $\mathcal{A}(\boldsymbol{\alpha}, \mathbf{y})$. Suppose $t \le r/2$ errors have occurred in locations $X_1 = \alpha_{i_1}, X_2 = \alpha_{i_2}, \ldots, X_t = \alpha_{i_t}$ with error values $Y_1 = a_{i_1}, Y_2 = a_{i_2}, \ldots, Y_t = a_{i_t}$. Then the error vector is $\mathbf{e} = (e_1, e_2, \ldots, e_n)$ with $e_j = 0$ for $j \neq X_1, X_2, \ldots, X_t$ and $e_{X_1} = a_{i_1} = Y_1, e_{X_2} = a_{i_2} = Y_2, \ldots, e_{X_t} = a_{i_t} = Y_t$. The decoding procedure completes in the following three steps- 1. Find the syndrome, 2. Find the error locator and error evaluator polynomials and 3. Find the locations, values of the errors and correct them, which are briefed below.

1. **Find the syndrome** : The Syndrome S is computed as $S = H\mathbf{e}^T$; i.e.

$$S = \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ \vdots \\ S_{r-1} \end{bmatrix} = \begin{bmatrix} y_1 & y_2 & \cdots & y_n \\ \alpha_1 y_1 & \alpha_2 y_2 & \cdots & \alpha_n y_n \\ \alpha_1^2 y_1 & \alpha_2^2 y_2 & \cdots & \alpha_n^2 y_n \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_1^{r-1} y_1 & \alpha_2^{r-1} y_2 & \cdots & \alpha_n^{r-1} y_n \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_n \end{bmatrix}$$

where $S_\mu = \sum\limits_{i=1}^{n} \alpha_i^\mu y_i e_i = \sum\limits_{\nu=1}^{t} \alpha_{i_\nu}^\mu y_{i_\nu} e_{X_\nu} = \sum\limits_{\nu=1}^{t} \alpha_{i_\nu}^\mu a_{i_\nu} y_{i_\nu} = \sum\limits_{\nu=1}^{t} X_\nu^\mu Y_\nu y_{i_\nu}$ for $\mu = 0, 1, \ldots, r-1$.

2. **Find the error locator and error evaluator polynomials** : The error locator polynomial

$$\sigma(z) = \prod_{i=1}^{t}(1 - X_i z) = \sum_{i=0}^{t} \sigma_i z^i \text{ with } \sigma_0 = 1 = \sigma(0)$$

has reciprocal of error locations X_i at its zeros and the error evaluator polynomial is

$$\omega(z) = \sum_{\nu=1}^{t} Y_\nu y_{i_\nu} \prod_{\mu=1,\mu\neq\nu}^{t} (1 - X_\mu z).$$

Note that the polynomials $\omega(z)$ and $\sigma(z)$ are related by

$$\frac{\omega(z)}{\sigma(z)} = S(z) \mod z^r$$

where $S(z) = \sum\limits_{\mu=0}^{r-1} S_\mu z^\mu$ as

$$\begin{aligned}
\frac{\omega(z)}{\sigma(z)} &= \frac{Y_1 y_{i_1}}{1 - X_1 z} + \frac{Y_2 y_{i_2}}{1 - X_2 z} + \cdots + \frac{Y_t y_{i_t}}{1 - X_t z} \\
&= \left(Y_1 y_{i_1} \sum_{\mu=0}^{r-1} X_1^\mu z^\mu + Y_2 y_{i_2} \sum_{\mu=0}^{r-1} X_2^\mu z^\mu + \cdots + Y_t y_{i_t} \sum_{\mu=0}^{r-1} X_t^\mu z^\mu \right) \mod z^r \\
&= \sum_{\mu=0}^{r-1} z^\mu \sum_{\nu=1}^{t} X_\nu^\mu Y_\nu y_{i_\nu} \mod z^r \\
&= \sum_{\mu=0}^{r-1} z^\mu S_\mu \mod z^r \\
&= S(z) \mod z^r
\end{aligned}$$

Employ the extended Euclidean algorithm to find $\omega(z)$, $\sigma(z)$ as follows :

(i) Set $r_{-1}(z) = z^r$, $r_0(z) = S(z)$. Define the polynomials $U_i(z)$, $V_i(z)$ recursively as

$$U_{-1}(z) = 0, U_0(z) = 1, V_{-1}(z) = 1, V_0(z) = 0$$

$$U_i(z) = q_i(z)U_{i-1}(z) + U_{i-2}(Z)$$

$$V_i(z) = q_i(z)V_{i-1}(z) + V_{i-2}(Z)$$

Then

$$\begin{aligned}
\begin{bmatrix} U_i(z) & U_{i-1}(z) \\ V_i(z) & V_{i-1}(z) \end{bmatrix} &= \begin{bmatrix} U_{i-1}(z) & U_{i-2}(z) \\ V_{i-1}(z) & V_{i-2}(z) \end{bmatrix} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} U_{i-2}(z) & U_{i-3}(z) \\ V_{i-2}(z) & V_{i-3}(z) \end{bmatrix} \begin{bmatrix} q_{i-1}(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} q_1(z) & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix}
\end{aligned}$$

Also define the polynomials $r_i(z)$ recursively as

$$\begin{aligned}
\begin{bmatrix} r_{i-2}(z) \\ r_{i-1}(z) \end{bmatrix} &= \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_{i-1}(z) \\ r_i(z) \end{bmatrix} \\
\begin{bmatrix} r_{i-3}(z) \\ r_{i-2}(z) \end{bmatrix} &= \begin{bmatrix} q_{i-1}(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_{i-1}(z) \\ r_i(z) \end{bmatrix} \\
&\vdots \\
\begin{bmatrix} r_{-1}(z) \\ r_0(z) \end{bmatrix} &= \begin{bmatrix} q_1(z) & 1 \\ 1 & 0 \end{bmatrix} \cdots \begin{bmatrix} q_{i-1}(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r_{i-1}(z) \\ r_i(z) \end{bmatrix} \\
&= \begin{bmatrix} U_i(z) & U_{i-1}(z) \\ V_i(z) & V_{i-1}(z) \end{bmatrix} \begin{bmatrix} r_{i-1}(z) \\ r_i(z) \end{bmatrix}
\end{aligned}$$

Note that

$$\begin{vmatrix} U_i(z) & U_{i-1}(z) \\ V_i(z) & V_{i-1}(z) \end{vmatrix} = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} \begin{vmatrix} q_1(z) & 1 \\ 1 & 0 \end{vmatrix} \cdots \begin{vmatrix} q_i(z) & 1 \\ 1 & 0 \end{vmatrix} = (-1)^i$$

which implies

$$\begin{bmatrix} r_{i-1}(z) \\ r_i(z) \end{bmatrix} = (-1)^i \begin{bmatrix} V_{i-1}(z) & -U_{i-1}(z) \\ -V_i(z) & U_i(z) \end{bmatrix} \begin{bmatrix} r_{-1}(z) \\ r_0(z) \end{bmatrix}$$

In particular,

$$\begin{aligned}
r_i(z) &= (-1)^i[-V_i(z)r_{-1}(z) + U_i(Z)r_0(z)] \\
&= (-1)^i U_i(Z) r_0(z) \mod (r_{-1}(z))
\end{aligned}$$

Let $\mathsf{deg}\ f$ denotes the degree of a polynomial f. Here, we have

$$\begin{aligned}
&\mathsf{deg}\ U_i = \sum_{k=1}^{i} \mathsf{deg}\ q_k = \mathsf{deg}\ V_i \\
&\mathsf{deg}\ r_{i-1} = \mathsf{deg}\ r_{-1} - \sum_{k=1}^{i} \mathsf{deg}\ q_k \\
&\mathsf{deg}\ U_i = \mathsf{deg}\ V_i = \mathsf{deg}\ r_{-1} - \mathsf{deg}\ r_{i-1} < \mathsf{deg}\ r_{-1}
\end{aligned}$$

(ii) Continue until reaching an $r_k(z)$ such that $\mathsf{deg}\ r_{k-1}(z) \geq r/2$ and $\mathsf{deg}\ r_k(z) \leq r/2 - 1$. Note that

$$\begin{aligned}
r_k(z) &= (-1)^k U_k(z) r_0(z) \mod r_{-1}(z) \\
&= (-1)^k U_k(z) S(z) \mod z^r
\end{aligned}$$

which implies that

$$\frac{(-1)^k r_k(z)}{U_k(z)} = S(z) \mod z^r$$

(iii) Set

$$\omega(z) = (-1)^k \delta r_k(z)$$
$$\sigma(z) = \delta U_k(z)$$

where δ is constant chosen to make $\sigma(0) = 1$. Observe that

$$\omega(z) = \sigma(z) S(z) \mod z^r$$

with deg $\sigma(z) \leq r/2$ and deg $\omega(z) \leq r/2 - 1$.

3. **Find the locations, values of the errors and correct them** : The error locators X_i are found as the reciprocals of the roots of $\sigma(z)$ and the error values are given by

$$Y_\nu = \frac{\omega(X_\nu^{-1})}{y_{i_\nu} \prod_{\mu \neq \nu} (1 - X_\mu X_\nu^{-1})}$$

as

$$\omega(X_\nu^{-1}) = \sum_{\nu=1}^{t} Y_\nu y_{i_\nu} \prod_{\mu \neq \nu,\ \mu=1}^{t} (1 - X_\mu X_\nu^{-1})$$
$$= Y_\nu y_{i_\nu} \prod_{\mu \neq \nu,\ \mu=1}^{t} (1 - X_\mu X_\nu^{-1})$$

3 Our KEM Protocol

We construct a key encapsulation mechanism $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ following the specifications of Definition 5.

- $\mathsf{KEM.Setup}(1^\lambda) \longrightarrow \mathsf{param}$: Taking security parameter λ as input, a trusted authority proceeds as follows to generate the global public parameters param.

(i) Sample $n_0, p_1, p_2, m \in \mathbb{Z}^+$, set $q = 2^{p_1},\ s = 2^{p_2}$ and $n = n_0 s < q^m$.
(ii) Select $t \in \mathbb{Z}^+$ such that $mst < n$. Set $w \leq st/2$ and $k = n - mst$.
(iii) Sample $k' \in \mathbb{Z}^+$ with $k' < k$.
(iv) Select three cryptographically secure hash functions $\mathcal{G} : (\mathsf{GF}(q))^{k'} \longrightarrow (\mathsf{GF}(q))^k$, $\mathcal{H} : (\mathsf{GF}(q))^{k'} \longrightarrow (\mathsf{GF}(q))^{k'}$ and $\mathcal{H}' : \{0,1\}^* \longrightarrow \{0,1\}^r$ where $r \in \mathbb{Z}^+$ denotes the desired key length.
(v) Publish the global parameters $\mathsf{param} = (n, n_0, k, k', w, q, s, t, r, m, \mathcal{G}, \mathcal{H}, \mathcal{H}')$.

- $\mathsf{KEM.KeyGen}(\mathsf{param}) \longrightarrow (\mathsf{pk}, \mathsf{sk})$: A user on input param, performs the following steps to generate the public key pk and secret key sk.

(i) Generate dyadic signature $\mathbf{h} = (h_0, h_1, \ldots, h_{n-1})$ using Algorithm 1 where $h_i \in \mathsf{GF}(q^m)$ for $i = 0, 1, \ldots, n-1$.
(ii) Select $\omega \overset{U}{\longleftarrow} \mathsf{GF}(q^m)$ with $\omega \neq \frac{1}{h_j} + \frac{1}{h_0}$, $j = 0, 1, \ldots, n-1$ and compute

$$u_i = \frac{1}{h_i} + \omega, \quad i = 0, 1, \ldots, s-1$$

$$v_j = \frac{1}{h_j} + \frac{1}{h_0} + \omega,\ j = 0, 1, \ldots, n-1.$$

Set $\mathbf{u} = (u_0, u_1, \ldots, u_{s-1})$ and $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$.

(iii) Construct $st \times n$ quasi-dyadic matrix $A = \begin{bmatrix} A_1 & A_2 & \cdots & A_t \end{bmatrix}^T$ where

$$A_i = \begin{bmatrix} \frac{1}{(u_0-v_0)^i} & \frac{1}{(u_0-v_1)^i} & \cdots & \frac{1}{(u_0-v_{n-1})^i} \\ \frac{1}{(u_1-v_0)^i} & \frac{1}{(u_1-v_1)^i} & \cdots & \frac{1}{(u_1-v_{n-1})^i} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{1}{(u_{s-1}-v_0)^i} & \frac{1}{(u_{s-1}-v_1)^i} & \cdots & \frac{1}{(u_{s-1}-v_{n-1})^i} \end{bmatrix}$$
$$= \begin{bmatrix} \frac{1}{(v_0-u_0)^i} & \frac{1}{(v_1-u_0)^i} & \cdots & \frac{1}{(v_{n-1}-u_0)^i} \\ \frac{1}{(v_0-u_1)^i} & \frac{1}{(v_1-u_1)^i} & \cdots & \frac{1}{(v_{n-1}-u_1)^i} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{1}{(v_0-u_{s-1})^i} & \frac{1}{(v_1-u_{s-1})^i} & \cdots & \frac{1}{(v_{n-1}-u_{s-1})^i} \end{bmatrix}$$

is the $s \times n$ matrix block that can be written as

$$A_i = [\hat{A}_{i_1} | \hat{A}_{i_2} | \cdots | \hat{A}_{i_{n_0}}].$$

Each block $\hat{A}_{i_k}$ is an $s \times s$ dyadic matrix for $k = 1, 2, \ldots, n_0$. For instance, take the first block

$$\hat{A}_{i_1} = \begin{bmatrix} \frac{1}{(u_0-v_0)^i} & \frac{1}{(u_0-v_1)^i} & \cdots & \frac{1}{(u_0-v_{s-1})^i} \\ \frac{1}{(u_1-v_0)^i} & \frac{1}{(u_1-v_1)^i} & \cdots & \frac{1}{(u_1-v_{s-1})^i} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{1}{(u_{s-1}-v_0)^i} & \frac{1}{(u_{s-1}-v_1)^i} & \cdots & \frac{1}{(u_{s-1}-v_{s-1})^i} \end{bmatrix}$$

which is symmetric as

$$u_i - v_j = \frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0} = u_j - v_i$$

and dyadic of order s as the $s \times s$ matrix

$$\begin{bmatrix} \frac{1}{(u_0-v_0)} & \frac{1}{(u_0-v_1)} & \frac{1}{(u_0-v_2)} & \cdots & \frac{1}{(u_0-v_{s-1})} \\ \frac{1}{(u_1-v_0)} & \frac{1}{(u_1-v_1)} & \frac{1}{(u_1-v_2)} & \cdots & \frac{1}{(u_1-v_{s-1})} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{1}{(u_{s-1}-v_0)} & \frac{1}{(u_{s-1}-v_1)^i} & \frac{1}{(u_{s-1}-v_2)^i} & \cdots & \frac{1}{(u_{s-1}-v_{s-1})} \end{bmatrix}$$
$$= \begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_{s-1} \\ h_1 & h_0 & h_3 & \cdots & h_{s-2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ h_{s-1} & h_{s-2} & h_{s-3} & \cdots & h_0 \end{bmatrix}$$
$$= \begin{bmatrix} h_{0\oplus 0} & h_{0\oplus 1} & h_{0\oplus 2} & \cdots & h_{0\oplus(s-1)} \\ h_{1\oplus 0} & h_{1\oplus 1} & h_{1\oplus 2} & \cdots & h_{1\oplus(s-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ h_{(s-1)\oplus 0} & h_{(s-1)\oplus 1} & h_{(s-1)\oplus 2} & \cdots & h_{(s-1)\oplus(s-1)} \end{bmatrix}$$

can be derived from the first row of the block using the relation $\frac{1}{h_{i\oplus j}} = \frac{1}{h_i} + \frac{1}{h_j} + \frac{1}{h_0}$. Since the powering process acts on every single element, $\hat{A}_{i_1}$ preserves its dyadic structure.

(iv) Choose $z_{is} \xleftarrow{U} \mathsf{GF}(q^m)$, $i = 0, 1, \ldots, n_0 - 1$ and set $z_{is+p} = z_{is}$, $p = 0, 1, \ldots, s-1$. Also set

$$\begin{aligned}\mathbf{z} = (&z_{0s}, z_{0s+1}, \ldots, z_{0s+s-1}; z_{1s}, z_{1s+1}, \ldots, z_{1s+s-1}; \ldots; z_{(n_0-1)s}, z_{(n_0-1)s+1},\\ &\ldots, z_{(n_0-1)s+s-1})\\ = (&z_0, z_1, \ldots, z_{n-1}) \in (\mathsf{GF}(q^m))^n\end{aligned}$$

where $n = n_0 s$.

(v) Compute

$$y_j = \frac{z_j}{\prod_{i=0}^{s-1}(u_i - v_j)^t} \quad \text{for } j = 0, 1, \ldots, n-1$$

and set $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1}) \in (\mathsf{GF}(q^m))^n$.

(vi) Construct $st \times n$ matrix $B = \begin{bmatrix} B_1 & B_2 & \cdots & B_t \end{bmatrix}^T$ where

$$B_i = \begin{bmatrix} \frac{z_0}{(v_0-u_0)^i} & \frac{z_1}{(v_1-u_0)^i} & \cdots & \frac{z_{n-1}}{(v_{n-1}-u_0)^i} \\ \frac{z_0}{(v_0-u_1)^i} & \frac{z_1}{(v_1-u_1)^i} & \cdots & \frac{z_{n-1}}{(v_{n-1}-u_1)^i} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{z_0}{(v_0-u_{s-1})^i} & \frac{z_1}{(v_1-u_{s-1})^i} & \cdots & \frac{z_{n-1}}{(v_{n-1}-u_{s-1})^i} \end{bmatrix}$$

is $s \times n$ matrix block. Sample a permutation matrix P of order st and compute $st \times n$ matrix $\overline{B} = PB$. The matrix $\overline{B}$ is a parity-check matrix of the GS code equivalent to its parity check matrix given in Definition 13 in Subsection 2.4.

(vii) Project $\overline{B}$ onto $\mathsf{GF}(q)$ using the $\mathsf{co\text{-}trace}$ function to form a $mst \times n$ matrix C where $\mathsf{co\text{-}trace}$ function converts an element of $\mathsf{GF}(q^m)$ to an element of $\mathsf{GF}(q)$ with respect to a basis of $\mathsf{GF}(q^m)$ over $\mathsf{GF}(q)$. For $\mathbf{a} \in \mathsf{GF}(q^m)$, $\mathsf{co\text{-}trace}(\mathbf{a}) = (a_0, a_1, \ldots, a_{m-1}) \in (\mathsf{GF}(q))^m$ satisfying $<\mathbf{g}, \mathbf{a}>= a_0 + a_1 q + a_2 q^2 + \cdots + a_{m-1} q^{m-1}$ where $a_i \in \mathsf{GF}(q)$ and $\mathbf{g} = (1, q, q^2, \ldots, q^{m-1})$ is a basis of $\mathsf{GF}(q^m)$ over $\mathsf{GF}(q)$. Thus if $\overline{B} = (\overline{b}_{ij})$ where $\overline{b}_{ij} \in \mathsf{GF}(q^m)$, then $C = (c_{ij})$ is obtained from $\overline{B}$ by replacing $\overline{b}_{ij}$ by $\mathsf{co\text{-}trace}(\overline{b}_{ij})$. Write the matrix C in the systematic form $(M|I_{n-k})$ where M is $(n-k) \times k$ matrix with $k = n - mst$. Note that the powering process acts on every single element, the z_i are chosen to be equal s-wise and all the operations occurring during the row reducing are performed block by block in the ring of dyadic matrices over $\mathsf{GF}(q)$. Consequently, the dyadic structure is prevented in C and in particular in M. Let

$$M = \begin{bmatrix} M_{0,0} & M_{0,1} & \cdots & M_{0,\frac{k}{s}-1} \\ M_{1,0} & M_{1,1} & \cdots & M_{1,\frac{k}{s}-1} \\ \cdots & \cdots & \cdots & \cdots \\ M_{mt-1,0} & M_{mt-1,1} & \cdots & M_{mt-1,\frac{k}{s}-1} \end{bmatrix}$$

where each block matrix $M_{i,j}$ is $s \times s$ dyadic matrix with dyadic signature $\boldsymbol{\psi}_{i,j} \in (\mathsf{GF}(q))^s$ which is the first row of $M_{i,j}$, $i = 0, 1, \ldots, mt-1$, $j = 0, 1, \ldots, \frac{k}{s} - 1$.

(viii) Publish the public key $\mathsf{pk} = \{\boldsymbol{\psi}_{i,j} \mid i = 0, 1, \ldots, mt-1,\ j = 0, 1, \ldots, \frac{k}{s} - 1\}$ and keep the secret key $\mathsf{sk} = (\mathbf{v}, \mathbf{y})$ to itself.

- $\mathsf{KEM.Encaps}(\mathsf{param}, \mathsf{pk}) \longrightarrow (\mathsf{CT}, K)$: Given system parameters param and public key pk, an encapsulator proceeds as follows to generate a ciphertext header $\mathsf{CT} \in (\mathsf{GF}(q))^{n-k+k'}$ and an encapsulation key $K \in \{0,1\}^r$.

(i) Sample $\mathbf{m} \xleftarrow{U} (\mathsf{GF}(q))^{k'}$ and compute $\mathbf{r} = \mathcal{G}(\mathbf{m}) \in (\mathsf{GF}(q))^k$, $\mathbf{d} = \mathcal{H}(\mathbf{m}) \in (\mathsf{GF}(q))^{k'}$ where $\mathcal{G}$ and $\mathcal{H}$ are the hash functions given in param.

(ii) Parse $\mathbf{r}$ as $\mathbf{r} = (\boldsymbol{\rho}||\boldsymbol{\sigma})$ where $\boldsymbol{\rho} \in (\mathsf{GF}(q))^{k-k'}$, $\boldsymbol{\sigma} \in (\mathsf{GF}(q))^{k'}$. Set $\boldsymbol{\mu} = (\boldsymbol{\rho}||\mathbf{m}) \in (\mathsf{GF}(q))^k$.

(iii) Run Algorithm 2 to generate a unique error vector $\mathbf{e}$ of length $n-k$ and weight $w - \mathsf{wt}(\boldsymbol{\mu})$ using $\boldsymbol{\sigma}$ as a seed. Note that Algorithm 2 uses an expansion function [1]. Set $\mathbf{e}' = (\mathbf{e}||\boldsymbol{\mu}) \in (\mathsf{GF}(q))^n$.

(iv) Using the public key pk=$\{\boldsymbol{\psi}_{i,j} \mid i = 0, 1, \ldots, mt-1,\ j = 0, 1, \ldots, \frac{k}{s} - 1\}$, compute $s \times s$ dyadic matrix $M_{i,j}$ with signature $\boldsymbol{\psi}_{i,j} \in (\mathsf{GF}(q))^s$ and reconstruct the parity check matrix $H = (M|I_{n-k})$ for the the GS code where

$$M = \begin{bmatrix} M_{0,0} & M_{0,1} & \cdots & M_{0,\frac{k}{s}-1} \\ M_{1,0} & M_{1,1} & \cdots & M_{1,\frac{k}{s}-1} \\ \cdots & \cdots & \cdots & \cdots \\ M_{mt-1,0} & M_{mt-1,1} & \cdots & M_{mt-1,\frac{k}{s}-1} \end{bmatrix}$$

and $n - k = mst$.

(v) Compute the syndrome $\mathbf{c} = H(\mathbf{e}')^T$ and the encapsulation key $K = \mathcal{H}'(\mathbf{m})$ where $\mathcal{H}'$ is the hash function given in param.

(vi) Publish the ciphertext header $\mathsf{CT} = (\mathbf{c}, \mathbf{d})$ and keep K as secret.

- $\mathsf{KEM.Decaps}(\mathsf{param}, \mathsf{sk}, \mathsf{CT}) \longrightarrow K$: On receiving a ciphertext header $\mathsf{CT} = (\mathbf{c}, \mathbf{d})$, a decapsulator executes the following steps using public parameters param and its secret key $\mathsf{sk} = (\mathbf{v}, \mathbf{y})$ where $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ and $\mathbf{y} = (y_0, y_1, \ldots, y_{n-1})$.

(i) First proceed as follows to decode $\mathbf{c}$ and find error vector $\mathbf{e}''$ of length n and weight w :

[1] For example, kangaroo twelve function [52] can be used as an expansion function.

Algorithm 2 Error vector derivation

```
Input: $q$, $n$, a seed $\bar{\mathbf{s}} = (\bar{s}_0, \bar{s}_1, \dots, \bar{s}_{k-1}) \in (\mathsf{GF}(q))^k$ of length $k$, a weight $w$,
    a function $\mathcal{F} : \mathsf{GF}(q) \longrightarrow \mathbb{Z}^+$.
Output: An error vector $\mathbf{e}$ of length $n$ and weight $w$.
1: $\mathbf{s} = (s_0, s_1, \dots, s_{n-1})$=Expand($\bar{\mathbf{s}}$); // Expand is an expansion function
2: $j = 0$; temp $= 0$; $d = 0$;
3: $\mathbf{e} = \mathbf{0}$; $\mathbf{v} = \mathbf{0}$;
4: for ($i = 0$ to $n-1$) do
5:     if ($s_i \bmod q \neq 0$) then
6:         if ($j = w$) then
7:             break;
8:         end if
9:         temp $= \mathcal{F}(s_d) \bmod n$;
10:        $d = d + 1$;
11:        for ($\nu = 0$ to $j$) do
12:            if (temp $= v_\nu$) then
13:                goto step 9;
14:            end if
15:        end for
16:        $v_j =$ temp;
17:        $e_{\mathsf{temp}} = s_i \bmod q$;
18:        temp $= 0$;
19:        $j = j + 1$;
20:    end if
21: end for
22: return $\mathbf{e} = (e_0, e_1, \dots, e_{n-1})$
```

(a) Use $\mathsf{sk} = (\mathbf{v}, \mathbf{y})$ to form $st \times n$ matrix

$$H' = \begin{bmatrix} y_0 & y_1 & \cdots & y_{n-1} \\ v_0 y_0 & v_1 y_1 & \cdots & v_{n-1} y_{n-1} \\ v_0^2 y_0 & v_1^2 y_1 & \cdots & v_{n-1}^2 y_{n-1} \\ \cdots & \cdots & \cdots & \cdots \\ v_0^{st-1} y_0 & v_1^{st-1} y_1 & \cdots & v_{n-1}^{st-1} y_{n-1} \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 1 & \cdots & 1 \\ v_0 & v_1 & \cdots & v_{n-1} \\ v_0^2 & v_1^2 & \cdots & v_{n-1}^2 \\ \cdots & \cdots & \cdots & \cdots \\ v_0^{st-1} & v_1^{st-1} & \cdots & v_{n-1}^{st-1} \end{bmatrix} \begin{bmatrix} y_0 & 0 & \cdots & 0 \\ 0 & y_1 & \cdots & 0 \\ 0 & 0 & y_2 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & y_{n-1} \end{bmatrix}.$$

Note that the $st \times n$ matrix H' is a parity check matrix in alternant form of the GS code over $\mathsf{GF}(q^m)$ whereas the matrix $H = [M|I_{(n-k)}]$ constructed during $\mathsf{KEM.KeyGen}$ or $\mathsf{KEM.Encaps}$ is a parity check matrix in the systematic form of the GS code over $\mathsf{GF}(q)$.

(b) As the GS code is an Alternant code, the parity check matrix H' is used to decode $\mathbf{c}$ by first computing the syndrome $S = H'(\mathbf{c}||\mathbf{0})^T$ where $\mathbf{0}$ represents the vector $(0, 0, \dots, 0)$ of length k and then by running algorithm described in Subsection 2.5 for the Alternant code to find the error locator polynomial

$$\omega(z) = \sum_{\nu=1}^{w} Y_\nu y_{i_\nu} \prod_{\mu=1, \mu \neq \nu}^{w} (1 - X_\mu z)$$

and error evaluator polynomial

$$\sigma(z) = \prod_{i=1}^{w}(1 - X_i z).$$

Let $X_1 = v_{i_1}, X_2 = v_{i_2}, \ldots, X_w = v_{i_w}$ be the error locations and $Y_1 = e_{X_1}, Y_2 = e_{X_2}, \ldots, Y_w = e_{X_w}$ be the error values.

(c) Set $\mathbf{e}'' = (e_1, e_2, \ldots, e_n)$ with $e_j = \begin{cases} 0 & \text{if } j \neq X_i, 1 \leq i \leq w \\ Y_i & \text{if } j = X_i, 1 \leq i \leq w \end{cases}$.

(ii) Let $\mathbf{e}'' = (\mathbf{e}_0 || \boldsymbol{\mu}') \in (\mathsf{GF}(q))^n$ and $\boldsymbol{\mu}' = (\boldsymbol{\rho}' || \mathbf{m}') \in (\mathsf{GF}(q))^k$ where $\mathbf{e}_0 \in (\mathsf{GF}(q))^{n-k}$, $\boldsymbol{\rho}' \in (\mathsf{GF}(q))^{k-k'}$, $\mathbf{m}' \in (\mathsf{GF}(q))^{k'}$.

(iii) Compute $\mathbf{r}' = \mathcal{G}(\mathbf{m}') \in (\mathsf{GF}(q))^k$ and $\mathbf{d}' = \mathcal{H}(\mathbf{m}') \in (\mathsf{GF}(q))^{k'}$ where $\mathcal{G}$ and $\mathcal{H}$ are the hash functions given in param.

(iv) Parse $\mathbf{r}'$ as $\mathbf{r}' = (\boldsymbol{\rho}'' || \boldsymbol{\sigma}')$ where $\boldsymbol{\rho}'' \in (\mathsf{GF}(q))^{k-k'}$, $\boldsymbol{\sigma}' \in (\mathsf{GF}(q))^{k'}$.

(v) Run Algorithm 2 to generate deterministically an error vector $\mathbf{e}_0'$ of length $n - k$ and weight $w - \mathsf{wt}(\boldsymbol{\mu}')$ using $\boldsymbol{\sigma}'$ as seed.

(vi) If $(\mathbf{e}_0 \neq \mathbf{e}_0') \vee (\boldsymbol{\rho}' \neq \boldsymbol{\rho}'') \vee (\mathbf{d} \neq \mathbf{d}')$, output $\perp$ indicating decapsulation failure. Otherwise, compute the encapsulation key $K = \mathcal{H}'(\mathbf{m}')$ where $\mathcal{H}'$ is the hash function given in param.

Correctness: While decoding $\mathbf{c}$, we form a parity check $st \times n$ matrix H' over $\mathsf{GF}(q^m)$ using the secret key $\mathsf{sk} = (\mathbf{v}, \mathbf{y})$ and find the syndrome $H'(\mathbf{c}||\mathbf{0})^T$ to estimate the error vector $\mathbf{e}'' \in (\mathsf{GF}(q))^n$ with $\mathsf{wt}(\mathbf{e}'') = w$. Note that, the ciphertext component $\mathbf{c} = H(\mathbf{e}')^T$ is the syndrome of $\mathbf{e}'$ where the matrix H is a parity check matrix in the systemetic form over $\mathsf{GF}(q)$ which is indistinguishable from a random matrix over $\mathsf{GF}(q)$. At the time of decoding $\mathbf{c}$, we need a parity check matrix in alternant form over $\mathsf{GF}(q^m)$. The parity check matrix H, a parity check matrix of GS code in the systemetic form derived from the public key pk, does not help to decode $\mathbf{c}$ as the SD problem is hard over $\mathsf{GF}(q)$. The decoding algorithm in our decapsulation procedure uses the parity check matrix H' (derived from the secret key sk) which is in alternant form over $\mathsf{GF}(q^m)$. This procedure can correct upto $st/2$ errors. In our scheme, the error vector $\mathbf{e}'$ used in the procedure $\mathsf{KEM.Encaps}$ satisfies $\mathsf{wt}(\mathbf{e}') = w \leq st/2$. Consequently, the decoding procedure will recover the correct $\mathbf{e}'$. We regenerate $\mathbf{e}_0'$ and $\boldsymbol{\rho}''$ and compare it with $\mathbf{e}_0$ and $\boldsymbol{\rho}'$ obtained after decoding. Since the error vector generation uses a deterministic function to get a fixed low weight error vector, $\mathbf{e}_0 = \mathbf{e}_0'$ and $\boldsymbol{\rho}' = \boldsymbol{\rho}''$ occurs.

4 Security

Theorem 3. *If the public key encryption scheme* $\mathsf{PKE}_1 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *described in Figure 2 is OW-VA secure (Definition 4 in Subsection 2.1) and there exist cryptographically secure hash functions, then the key encapsulation mechanism* $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *as described in Section 3 achieves* $\mathsf{IND\text{-}CCA}$ *security (Definition 6 in Subsection 2.2) when the hash function* $\mathcal{H}'$ *is modeled as a random oracle.*

- $\mathsf{PKE}_1.\mathsf{Setup}(1^\lambda) \longrightarrow \mathsf{param}$: A trusted authority runs $\mathsf{KEM.Setup}(1^\lambda)$ to get global parameters $\mathsf{param} = (n, n_0, k, k', w, q, s, t, r, m, \mathcal{G}, \mathcal{H}, \mathcal{H}')$ taking security parameter λ as input.

- $\mathsf{PKE}_1.\mathsf{KeyGen}(\mathsf{param}) \longrightarrow (\mathsf{pk}, \mathsf{sk})$: A user generates public-secret key pair $(\mathsf{pk}, \mathsf{sk})$ by running $\mathsf{KEM.KeyGen}(\mathsf{param})$ where $\mathsf{pk} = \{\boldsymbol{\psi}_{i,j} \mid i = 0, 1, \ldots, mt-1,\ j = 0, 1, \ldots, \frac{k}{s}-1\}$, $\boldsymbol{\psi}_{i,j} \in (\mathsf{GF}(q))^s$ and $\mathsf{sk} = (\mathbf{v}, \mathbf{y})$.

- $\mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathsf{CT}$: An encryptor encrypts a message $\mathbf{m} \in \mathcal{M} = (\mathsf{GF}(q))^{k'}$ and produces a ciphertext CT as follows.

  1. Compute $\mathbf{r} = \mathcal{G}(\mathbf{m}) \in (\mathsf{GF}(q))^k$, $\mathbf{d} = \mathcal{H}(\mathbf{m}) \in (\mathsf{GF}(q))^{k'}$.
  2. Parse $\mathbf{r} = (\boldsymbol{\rho}||\boldsymbol{\sigma})$ where $\boldsymbol{\rho} \in (\mathsf{GF}(q))^{k-k'}$, $\boldsymbol{\sigma} \in (\mathsf{GF}(q))^{k'}$.
  3. Set $\boldsymbol{\mu} = (\boldsymbol{\rho}||\mathbf{m}) \in (\mathsf{GF}(q))^k$.
  4. Run Algorithm 2 using $\boldsymbol{\sigma}$ as a seed to obtain an error vector $\mathbf{e}$ of length $n-k$ and weight $w - \mathsf{wt}(\boldsymbol{\mu})$ and set $\mathbf{e}' = (\mathbf{e}||\boldsymbol{\mu}) \in (\mathsf{GF}(q))^n$.
  5. Use the public key $\mathsf{pk} = \{\boldsymbol{\psi}_{i,j} | i = 0, 1, \ldots, mt-1,\ j = 0, 1, \ldots, \frac{k}{s}-1\}$ as in $\mathsf{KEM.Encaps}(\mathsf{param},\mathsf{pk})$ and construct the matrix $H_{(n-k)\times n} = (M|I_{n-k})$ where $M = (M_{i,j})$, $M_{i,j}$ is a $s \times s$ dyadic matrix with signature $\boldsymbol{\psi}_{i,j}$, $i = 0, 1, \ldots, mt-1$, $j = 0, 1, \ldots, \frac{k}{s}-1$.
  6. Compute $\mathbf{c} = H(\mathbf{e}')^T$.
  7. Return the ciphertext $\mathsf{CT} = (\mathbf{c},\ \mathbf{d}) \in \mathcal{C} = (\mathsf{GF}(q))^{n-k+k'}$.

- $\mathsf{PKE}_1.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT}) \longrightarrow \mathbf{m}'$: On receiving the ciphertext CT, the decryptor executes the following steps using public parameters param and its secret key $\mathsf{sk} = (\mathbf{v}, \mathbf{y})$.

  1. Use the secret key $\mathsf{sk} = (\mathbf{v}, \mathbf{y})$ to form a parity check matrix H' as in the procedure $\mathsf{KEM.Decaps}(\mathsf{param},\mathsf{sk},\mathsf{CT})$.
  2. To decode $\mathbf{c}$ (extracted from CT), find error $\mathbf{e}''$ of weight w and length n by running the decoding algorithm for Alternant codes in Subsection 2.5 with syndrome $H'(\mathbf{c}||\mathbf{0})^T$.
  3. Parse $\mathbf{e}'' = (\mathbf{e}_0||\boldsymbol{\mu}') \in (\mathsf{GF}(q))^n$ and $\boldsymbol{\mu}'=(\boldsymbol{\rho}'||\mathbf{m}') \in (\mathsf{GF}(q))^k$ where $\mathbf{e_0} \in (\mathsf{GF}(q))^{n-k}$, $\boldsymbol{\rho}' \in (\mathsf{GF}(q))^{k-k'}$, $\mathbf{m}' \in (\mathsf{GF}(q))^{k'}$.
  4. Compute $\mathbf{r}' = \mathcal{G}(\mathbf{m}') \in (\mathsf{GF}(q))^k$ and $\mathbf{d}' = \mathcal{H}(\mathbf{m}') \in (\mathsf{GF}(q))^{k'}$.
  5. Parse $\mathbf{r}' = (\boldsymbol{\rho}''||\boldsymbol{\sigma}')$ where $\boldsymbol{\rho}'' \in (\mathsf{GF}(q))^{k-k'}$, $\boldsymbol{\sigma}' \in (\mathsf{GF}(q))^{k'}$.
  6. Generate error vector $\mathbf{e}_0'$ of length $n-k$ and weight $w - \mathsf{wt}(\boldsymbol{\mu}')$ by running Algorithm 2 with $\boldsymbol{\sigma}'$ as seed.
  7. If $(\mathbf{e}_0 \neq \mathbf{e}_0') \vee (\boldsymbol{\rho}' \neq \boldsymbol{\rho}'') \vee (\mathbf{d} \neq \mathbf{d}')$, output $\perp$ indicating decryption failure. Otherwise, return $\mathbf{m}'$.

Fig. 2. Scheme $\mathsf{PKE}_1 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$

Proof. Let $\mathcal{B}$ be a PPT adversary against the IND-CCA security of KEM providing at most n_D queries to KEM.Decaps oracle and at most $n_{\mathcal{H}'}$ queries to the hash oracle $\mathcal{H}'$. We show that $\exists$ a PPT adversary $\mathcal{A}$ against the OW-VA security of the scheme PKE_1. We start with a sequence of games and the view of the adversary $\mathcal{B}$ is shown to be computationally indistinguishable in any of the consecutive games. Finally, we end up in a game that statistically hides the challenge bit as required. All the games are defined in Figure 3 and Figure 4. Let E_j be the event that $b = b'$ in game $\mathbf{G}_j$, $j = 0, 1, 2, 3$.

Game $\mathbf{G}_0$: As usual, game $\mathbf{G}_0$ (Figure 3) is the standard IND-CCA security game (Definition 6 in Subsection 2.2) for the KEM and we have

$$|\Pr[E_0] - 1/2| = \mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA}}(\mathcal{B}).$$

Game $\mathbf{G}_1$: In game $\mathbf{G}_1$, a message $\mathbf{m}^*$ is chosen randomly and the ciphertext CT^* is computed by running $\mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}^*; \mathbf{r}^*)$. The challenger

- The challenger $\mathcal{S}$ generates $\mathsf{param} \longleftarrow \mathsf{KEM.Setup}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{KEM.KeyGen}(\mathsf{param})$ for a security parameter λ and sends $\mathsf{param}, \mathsf{pk}$ to $\mathcal{B}$.
- The PPT adversary $\mathcal{B}$ has access to the decapsulation oracle $\mathsf{KEM.Decaps}$ to which $\mathcal{B}$ can make polynomially many ciphertext queries CT_i and gets the corresponding key $K_i \in \mathcal{K} = \{0,1\}^r$ from $\mathcal{S}$.
- The challenger $\mathcal{S}$ picks a random bit b from $\{0,1\}$, runs $\mathsf{KEM.Encaps}(\mathsf{param}, \mathsf{pk})$ to generate a ciphertext-key pair (CT^*, K_0^*) with $\mathsf{CT}^* \neq \mathsf{CT}_i$, selects randomly $K_1^* \in \mathcal{K}$ and sends the pair (CT^*, K_b^*) to $\mathcal{B}$.
- The adversary $\mathcal{B}$ having the pair (CT^*, K_b^*) keeps performing polynomially many decapsulation queries on $\mathsf{CT}_i \neq \mathsf{CT}^*$ and outputs b'.

Fig. 3. Game $\mathbf{G}_0$ in the proof of Theorem 3

- The challenger $\mathcal{S}$ generates $\mathsf{param} \longleftarrow \mathsf{PKE}_1.\mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{PKE}_1.\mathsf{KeyGen}(\mathsf{param})$ for a security parameter λ and sends $\mathsf{param}, \mathsf{pk}$ to $\mathcal{B}$.
- The PPT adversary $\mathcal{B}$ has access to the decapsulation oracle Decaps (see Figure 5) to which $\mathcal{B}$ can make polynomially many ciphertext queries CT_i and gets the corresponding key $K_i \in \mathcal{K}$ from $\mathcal{S}$.
- The challenger $\mathcal{S}$ picks a random bit b from $\{0,1\}$, chooses a message $\mathbf{m}^* \xleftarrow{U} \mathcal{M}$, runs $\mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}^*; \mathbf{r}^*)$ to generate a ciphertext CT^*, computes $K_0^* = \mathcal{H}'(\mathbf{m}^*)$, selects randomly $K_1^* \in \mathcal{K}$ and sends the pair (CT^*, K_b^*) to $\mathcal{B}$.
- The adversary $\mathcal{B}$ having the pair (CT^*, K_b^*) keeps performing polynomially many decapsulation queries on $\mathsf{CT}_i \neq \mathsf{CT}^*$ to Decaps oracle and hash queries on $\mathbf{m}_i$ to hash oracle $\mathcal{H}'$and outputs b' (see Figure 5 for hash oracle $\mathcal{H}'$ and decapsulation oracle Decaps).

Fig. 4. Sequence of games $\mathbf{G}_j, j = 1, 2, 3$ in the proof of Theorem 3

$\mathcal{S}$ maintains a hash list $Q_{\mathcal{H}'}$ (initially empty) and records all entries of the form $(\mathbf{m}, K)$ where hash oracle $\mathcal{H}'$ is queried on some message $\mathbf{m} \in \mathcal{M}$. Note that both games $\mathbf{G}_0$ and $\mathbf{G}_1$ proceed identically and we get

$$\Pr[E_0] = \Pr[E_1].$$

Game $\mathbf{G}_2$: In game $\mathbf{G}_2$, the hash oracle $\mathcal{H}'$ and the decapsulation oracle Decaps are answered in such a way that they no longer make use of the secret key sk except for testing whether $\mathsf{PKE}_1.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT}) \in \mathcal{M}$ for a given ciphertext CT (line 12 of Decaps oracle in Figure 5). The hash list $Q_{\mathcal{H}'}$ records all entries of the form $(\mathbf{m}, K)$ where hash oracle $\mathcal{H}'$ is queried on some message $\mathbf{m} \in \mathcal{M}$. Another list Q_D stores entries of the form (CT, K) where either Decaps oracle is queried on some ciphertext CT or the hash oracle $\mathcal{H}'$ is queried on some message $\mathbf{m} \in \mathcal{M}$ satisfying $\mathsf{CT} \longleftarrow \mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r})$ with $\mathsf{PKE}_1.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT}) \longrightarrow \mathbf{m}$.
Let X denotes the event that a correctness error has occurred in the underlying PKE_1 scheme. More specifically, X is the event that either the list $Q_{\mathcal{H}'}$ contains an entry $(\mathbf{m}, K)$ with the condition $\mathsf{PKE}_1.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r})) \neq \mathbf{m}$ or the list Q_D contains an entry (CT, K) with the condition $\mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathsf{PKE}_1.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT}); \mathbf{r}) \neq \mathsf{CT}$ or both.
Claim : The view of $\mathcal{B}$ is identical in games $\mathbf{G}_1$ and $\mathbf{G}_2$ unless the event X occurs.
Proof of claim. To prove this, consider a fixed PKE_1 ciphertext CT (placed as a Decaps query) with $\mathbf{m} \longleftarrow \mathsf{PKE}_1.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT})$. Note that when $\mathbf{m} \notin \mathcal{M}$, the decapsulation oracle $\mathsf{Decaps}(\mathsf{CT})$ returns $\perp$ in both games $\mathbf{G}_1$ and $\mathbf{G}_2$. Sup-

```
$\mathcal{H}'(\mathbf{m})$

1. for the game $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3$ do
2.   if $\exists K$ such that $(\mathbf{m}, K) \in Q_{\mathcal{H}'}$
3.     return $K$;
4.   end if
5. $\mathsf{CT} = (\mathbf{c}, \mathbf{d}) \longleftarrow \mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r})$;
6. $K \overset{U}{\longleftarrow} \mathcal{K}$;
7. end for
8. for the game $\mathbf{G}_3$ do
9.   if $\mathbf{m} = \mathbf{m}^*$ and $\mathsf{CT}^*$ defined
10.     $Y = \mathsf{true}$;
11.     abort;
12.   end if
13. end for
14. for the game $\mathbf{G}_2, \mathbf{G}_3$ do
15.   if $\exists K'$ such that $(\mathsf{CT}, K') \in Q_D$
16.     $K = K'$;
17.   else
18.     $Q_D = Q_D \cup \{(\mathsf{CT}, K)\}$;
19.   end if
20. end for
21. for the game $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3$ do
22.   $Q_{\mathcal{H}'} = Q_{\mathcal{H}'} \cup \{(\mathbf{m}, K)\}$;
23.   return $K$;
24. end for
```

```
$\mathsf{Decaps}(\mathsf{CT} \neq \mathsf{CT}^*)$

1. for game $\mathbf{G}_1$ do
2.   $\mathbf{m}' \longleftarrow \mathsf{PKE}_1.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT})$;
3.   if $\mathbf{m}' = \perp$
4.     return $\perp$;
5.   end if
6.   return $K = \mathcal{H}'(\mathbf{m}')$;
7. end for
8. for games $\mathbf{G}_2, \mathbf{G}_3$ do
9.   if $\exists K$ such that $(\mathsf{CT}, K) \in Q_D$
10.     return $K$;
11.   end if
12.   if $\mathsf{PKE}_1.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT}) \notin \mathcal{M}$
13.     return $\perp$;
14.   end if
15.   $K \overset{U}{\longleftarrow} \mathcal{K}$ ;
16.   $Q_D = Q_D \cup \{(\mathsf{CT}, K)\}$;
17.   return $K$;
18. end for
```

Fig. 5. The hash oracles $\mathcal{H}'$ and the decapsulation oracle Decaps for games $\mathbf{G}_j, j = 1, 2, 3$ in the proof of Theorem 3

pose $\mathbf{m} \in \mathcal{M}$. We now show that in game $\mathbf{G}_2$, $\mathsf{Decaps}(\mathsf{CT}) \longrightarrow \mathcal{H}'(\mathbf{m})$ for the PKE_1 ciphertext CT of a message $\mathbf{m} \in \mathcal{M}$ with $\mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathsf{CT}$. We distinguish two cases – $\mathcal{B}$ queries hash oracle $\mathcal{H}'$ on $\mathbf{m}$ before making Decaps oracle on CT, or the other way round.

Case 1: Let the oracle $\mathcal{H}'$ be queried on $\mathbf{m}$ first by $\mathcal{B}$ before decapsulation query on PKE_1 ciphertext CT. Since Decaps oracle was not yet queried on CT, no entry of the form (CT, K) exist in the current list Q_D yet. Therefore, besides adding $(\mathbf{m}, K \overset{U}{\longleftarrow} \mathcal{K})$ to the list $Q_{\mathcal{H}'}$(line 22 of $\mathcal{H}'$ oracle in Figure 5), the challenger $\mathcal{S}$ also adds (CT, K) to the list Q_D (line 18 of $\mathcal{H}'$ oracle in Figure 5), thereby defining $\mathsf{Decaps}(\mathsf{CT}) \longrightarrow K = \mathcal{H}'(\mathbf{m})$.

Case 2: Let the oracle Decaps be queried on PKE_1 ciphertext CT before the hash oracle $\mathcal{H}'$ is queried on $\mathbf{m}$. Then no entry of the form (CT, K) exists in Q_D yet. Otherwise, $\mathcal{H}'$ already was queried on a message $\mathbf{m}'' \neq \mathbf{m}$ (because Decaps oracle is assumed to be queried first on CT and the oracle $\mathcal{H}'$ was not yet queried on $\mathbf{m}$) satisfying $\mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}''; \mathbf{r}'') \longrightarrow \mathsf{CT}$ with $\mathsf{PKE}_1.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT}) \longrightarrow \mathbf{m}''$. This is a contradiction to the fact that the same PKE_1 ciphertext CT is generated for two different messages $\mathbf{m}'', \mathbf{m}$ using randomness $\mathbf{r}, \mathbf{r}''$ respectively where $\mathbf{r} = \mathcal{G}(\mathbf{m}) \neq \mathcal{G}(\mathbf{m}'') = \mathbf{r}''$ for a cryptographically secure hash function $\mathcal{G}$. The randomness $\mathbf{r} = \mathcal{G}(\mathbf{m})$ used in the encryption algorithm of the PKE_1 is generated determinitically using the message $\mathbf{m}$ and the hash function $\mathcal{G}$. Consequently, two different messages $(\mathbf{m}' \neq \mathbf{m})$ can not yield the same ciphertext (CT). Therefore, Decaps oracle adds $(\mathsf{CT}, K \overset{U}{\longleftarrow} \mathcal{K})$ to the

list Q_D, thereby defining $\mathsf{Decaps}(\mathsf{CT}) \longrightarrow K$. When queried on $\mathbf{m}$ afterwards for hash oracle $\mathcal{H}'$, an entry of the form (CT, K) already exists in the list Q_D (line 15 of $\mathcal{H}'$ oracle in Figure 5). By adding $(\mathbf{m}, K)$ to the list $Q_{\mathcal{H}'}$ and returning K, the hash oracle $\mathcal{H}'$ defines $\mathcal{H}'(\mathbf{m}) = K \longleftarrow \mathsf{Decaps}(\mathsf{CT})$.
Hence, $\mathcal{B}$'s view is identical in games $\mathbf{G}_1$ and $\mathbf{G}_2$ unless a correctness error X occurs. ■ (of Claim)

As $\Pr[X] = 0$ for our KEM, we have

$$\Pr[E_1] = \Pr[E_2].$$

Game $\mathbf{G}_3$: In game $\mathbf{G}_3$, the challenger $\mathcal{S}$ sets a flag $Y = \mathsf{true}$ and aborts (with uniformly random output) immediately on the event when $\mathcal{B}$ queries the hash oracle $\mathcal{H}'$ on $\mathbf{m}^*$. Hence,

$$|\Pr[E_2] - \Pr[E_3]| \leq \Pr[Y = \mathsf{true}].$$

In game $\mathbf{G}_3$, $\mathcal{H}'(\mathbf{m}^*)$ will never be given to $\mathcal{B}$ neither through a query on hash oracle $\mathcal{H}'$ nor through a query on decapsulation oracle Decaps, meaning bit b is independent from $\mathcal{B}$'s view. Thus, $\Pr[E_3] = 1/2$. Now it remains to bound $\Pr[Y = \mathsf{true}]$. To this end, we construct an adversary $\mathcal{A}$ against the $\mathsf{OW\text{-}VA}$ security of PKE_1 simulating game $\mathbf{G}_3$ for $\mathcal{B}$ as in Figure 6. Here $\mathcal{B}$ uses Decaps

```
$\mathcal{A}^{\mathsf{CVO}(\cdot)}(\mathsf{param}, \mathsf{pk}, \mathsf{CT}^*)$

1. $K^* \xleftarrow{U} \mathcal{K}$;
2. $b' \longleftarrow \mathcal{B}^{\mathsf{Decaps}(\cdot), \mathcal{H}'(\cdot)}(\mathsf{param}, \mathsf{pk}, \mathsf{CT}^*, K^*)$;
3. if $\exists(\mathbf{m}', K') \in Q_{\mathcal{H}'}$ such that
        $\mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}'; \mathbf{r}) \longrightarrow \mathsf{CT}^*$
4.     return $\mathbf{m}'$;
5. else
6.     abort;
7. end if
```

```
$\mathsf{Decaps}(\mathsf{CT} \neq \mathsf{CT}^*)$

1. if $\exists K$ such that $(\mathsf{CT}, K) \in Q_D$
2.     return $K$;
3. end if
4. if $\mathsf{CVO}(\mathsf{CT}) = 0$
5.     return $\perp$;
6. end if
7. $K \xleftarrow{U} \mathcal{K}$;
8. $Q_D = Q_D \cup \{(\mathsf{CT}, K)\}$;
9. return $K$;
```

Fig. 6. Adversary $\mathcal{A}$ against $\mathsf{OW\text{-}VA}$ security of PKE_1

oracle given in Figure 6 with the same hash oracle $\mathcal{H}'$ for game $\mathbf{G}_2$ in Figure 5. The Ciphertext Validity Oracle $\mathsf{CVO}(\cdot)$ used in Figure 6 is given in Figure 1 which does the same work as in the Decaps oracle in Figure 4 for game $\mathbf{G}_3$. Consequently, the simulation is perfect until $Y = \mathsf{true}$ occurs. Furthermore, $Y = \mathsf{true}$ ensures that $\mathcal{B}$ has queried $\mathcal{H}'(\mathbf{m}^*)$, which implies that $(\mathbf{m}^*, K') \in Q_{\mathcal{H}'}$ for some $K' \in \mathcal{K}$ where the list $Q_{\mathcal{H}'}$ is maintained by the adversary $\mathcal{A}$ simulating G_3 for $\mathcal{B}$. In this case, we have $\mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}^*; \mathbf{r}^*) \longrightarrow \mathsf{CT}^*$ and hence $\mathcal{A}$ returns $\mathbf{m}^*$. Thus,

$$\Pr[Y = \mathsf{true}] = \mathsf{Adv}^{\mathsf{OW\text{-}VA}}_{\mathsf{PKE}_1}(\mathcal{A})$$

Combining all the probabilities, we get

$$\begin{aligned}\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{KEM}}(\mathcal{B}) &= |\Pr[E_0] - 1/2| \\ &= |\Pr[E_1] - 1/2| \\ &= |\Pr[E_2] - 1/2| \\ &= |\Pr[E_2] - \Pr[E_3]| \\ &\leq \Pr[Y = \text{true}] \\ &= \mathsf{Adv}^{\mathsf{OW\text{-}VA}}_{\mathsf{PKE}_1}(\mathcal{A})\end{aligned}$$

which completes our proof. ■

Theorem 4. *If the public key encryption scheme* PKE_2 =(Setup, KeyGen, Enc, Dec) *described in Figure 7 is* IND-CPA *secure (Definition 3 in Subsection 2.1), then the public key encryption scheme* $\mathsf{PKE}_1 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *as described in Figure 2 provides* OW-PCVA *security (Definition 4 in Subsection 2.1) when the hash function* $\mathcal{G}$ *is modeled as a random oracle.*

Proof. Let $\mathcal{A}$ be a PPT adversary against the OW-PCVA security of the scheme PKE_1 with at most $n_{\mathcal{G}}$ queries to the hash oracle $\mathcal{G}$, n_P queries to the oracle PCO (Figure 1) and n_V queries to oracle CVO (Figure 1). We show that we can construct a PPT adversary against the IND-CPA security of the scheme PKE_2. We first define the sequence of games $\mathbf{G}_j, j = 0, 1, 2, 3, 4$ in Figure 8 and Figure 9. The view of the PPT adversary $\mathcal{A}$ is shown to be computationally indistinguishable in any of the consecutive games. Let E_j be the event that $\mathbf{m}' = \mathbf{m}^*$ in game $\mathbf{G}_j$, $j = 0, 1, 2, 3, 4$.

Game $\mathbf{G}_0$: As game $\mathbf{G}_0$ (Figure 8) is the standard OW-PCVA game (Definition 4), we have

$$\Pr[E_0] = \mathsf{Adv}^{\mathsf{OW\text{-}PCVA}}_{\mathsf{PKE}_1}(\mathcal{A}).$$

Game $\mathbf{G}_1$: In this game, $\mathbf{c}^*$ is computed by running $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}^*; \mathbf{r}^*)$ for the message $\mathbf{m}^*$ and the oracles queries to PCO and CVO are answered as in Figure 10. When a query $(\mathbf{m}, \mathbf{c})$ is made to the oracle PCO, the challenger $\mathcal{S}$ decrypts $\mathbf{c}$ to recover $\mathbf{m}' \longleftarrow \mathsf{PKE}_2.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathbf{c})$ and returns 1 if $\mathbf{m}' = \mathbf{m}$ with $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}'; \mathcal{G}(\mathbf{m}')) \longrightarrow \mathbf{c}$. The condition $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}'; \mathcal{G}(\mathbf{m}')) \longrightarrow \mathbf{c}$ actually addresses the checking steps 3-5 in Figure 2 in $\mathsf{PKE}_1.\mathsf{Dec}$ $(\mathsf{param}, \mathsf{sk}, \mathsf{CT} = (\mathbf{c}, \mathbf{d}))$. On query $\mathbf{c} \neq \mathbf{c}^*$ to the oracle CVO, the challenger computes $\mathbf{m}' \longleftarrow \mathsf{PKE}_2.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathbf{c})$ and returns 1 if $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}'; \mathcal{G}(\mathbf{m}')) \longrightarrow \mathbf{c}$ with $\mathbf{m}' \in \mathcal{M}$. The challenger $\mathcal{S}$ maintains hash list $Q_{\mathcal{G}}$ (initially empty) to store random oracle queries made to the hash oracle $\mathcal{G}$ with the convention that $\mathbf{r} = \mathcal{G}(\mathbf{m})$ if and only if $(\mathbf{m}, \mathbf{r}) \in Q_{\mathcal{G}}$. Note that both the games $\mathbf{G}_0$ and $\mathbf{G}_1$ proceed identically. Hence, we have

$$\Pr[E_0] = \Pr[E_1].$$

- $\mathsf{PKE}_2.\mathsf{Setup}(1^\lambda) \longrightarrow \mathsf{param}$: A trusted authority takes security parameter λ as input and runs $\mathsf{PKE}_1.\mathsf{Setup}(1^\lambda)$ to get public parameters $\mathsf{param} = (n, n_0, k, k', w, q, s, t, r, m, \mathcal{G}, \mathcal{H}, \mathcal{H}')$.

- $\mathsf{PKE}_2.\mathsf{KeyGen}(\mathsf{param}) \longrightarrow (\mathsf{pk}, \mathsf{sk})$: A user generates the key pair $(\mathsf{pk}, \mathsf{sk})$ by running $\mathsf{PKE}_1.\mathsf{KeyGen}(\mathsf{param})$ where the public key $\mathsf{pk} = \{\boldsymbol{\psi}_{i,j} \mid i = 0, 1, \ldots, mt-1,\ j = 0, 1, \ldots, \frac{k}{s} - 1\}$, $\boldsymbol{\psi}_{i,j} \in (\mathsf{GF}(q))^s$ and the secret key $\mathsf{sk} = (\mathbf{v}, \mathbf{y})$.

- $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathbf{c}$: An encryptor encrypts a message $\mathbf{m} \in \mathcal{M} = (\mathsf{GF}(q))^{k'}$ and produces a ciphertext CT as follows.

  1. Compute $\mathbf{r} = \mathcal{G}(\mathbf{m}) \in (\mathsf{GF}(q))^k$, $\mathbf{d} = \mathcal{H}(\mathbf{m}) \in (\mathsf{GF}(q))^{k'}$.
  2. Parse $\mathbf{r} = (\boldsymbol{\rho}||\boldsymbol{\sigma})$ where $\boldsymbol{\rho} \in (\mathsf{GF}(q))^{k-k'}$, $\boldsymbol{\sigma} \in (\mathsf{GF}(q))^{k'}$.
  3. Set $\boldsymbol{\mu} = (\boldsymbol{\rho}||\mathbf{m}) \in (\mathsf{GF}(q))^k$.
  4. Run Algorithm 2 using $\boldsymbol{\sigma}$ as a seed to obtain an error vector $\mathbf{e}$ of length $n-k$ and weight $w - \mathsf{wt}(\boldsymbol{\mu})$ and set $\mathbf{e}' = (\mathbf{e}||\boldsymbol{\mu}) \in (\mathsf{GF}(q))^n$.
  5. Use the public key $\mathsf{pk} = \{\boldsymbol{\psi}_{i,j} | i = 0, 1, \ldots, mt-1,\ j = 0, 1, \ldots, \frac{k}{s} - 1\}$ as in $\mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r})$ and construct the matrix $H_{(n-k)\times n} = (M|I_{n-k})$ where $M = (M_{i,j})$, $M_{i,j}$ is a $s \times s$ dyadic matrix with signature $\boldsymbol{\psi}_{i,j}$, $i = 0, 1, \ldots, mt-1$, $j = 0, 1, \ldots, \frac{k}{s} - 1$.
  6. Compute $\mathbf{c} = H(\mathbf{e}')^T$.

- $\mathsf{PKE}_2.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathbf{c}) \longrightarrow \mathbf{m}'$: On receiving the ciphertext $\mathbf{c}$, the decryptor performs the following steps using public parameters param and its secret key $\mathsf{sk} = (\mathbf{v}, \mathbf{y})$.

  1. Use the secret key $\mathsf{sk} = (\mathbf{v}, \mathbf{y})$ to form a parity check matrix H' as in the procedure $\mathsf{PKE}_1.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathsf{CT})$
  2. To decode $\mathbf{c}$, find error $\mathbf{e}''$ of weight w and length n by running the decoding algorithm for Alternant codes in Subsection 2.5 with syndrome $H'(\mathbf{c}||\mathbf{0})^T$.
  3. Parse $\mathbf{e}'' = (\mathbf{e}_0||\boldsymbol{\mu}') \in (\mathsf{GF}(q))^n$ and $\boldsymbol{\mu}'=(\boldsymbol{\rho}'||\mathbf{m}') \in (\mathsf{GF}(q))^k$ where $\mathbf{e}_0 \in (\mathsf{GF}(q))^{n-k}$, $\boldsymbol{\rho}' \in (\mathsf{GF}(q))^{k-k'}$, $\mathbf{m}' \in (\mathsf{GF}(q))^{k'}$.
  4. Return $\mathbf{m}'$.

Fig. 7. Scheme $\mathsf{PKE}_2 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$

Game $\mathbf{G}_2$: In game $\mathbf{G}_2$, a query $c \neq c^*$ to the CVO oracle is responded by first decrypting $\mathbf{c}$ as with one which computes $\mathbf{m}' \longleftarrow \mathsf{PKE}_2.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathbf{c})$ and returning 1 if there exists a previous record $(\mathbf{m}, \mathbf{r}) \in Q_{\mathcal{G}}$ such that $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathbf{c}$ and $\mathbf{m} = \mathbf{m}'$.
Next we show that the scheme PKE_2 is γ-uniform.

Lemma 1. *The scheme* PKE_2 *is γ-uniform (Definition 2) with* $\gamma = \dfrac{q^{-(k-k')}}{\binom{n-k}{w-\mathsf{wt}(\boldsymbol{\mu})}}$.

Proof of Lemma 1. Let $\mathbf{c}$ be a generic vector of the PKE_2 ciphertext space $(\mathsf{GF}(q))^{n-k}$. Then either $\mathbf{c}$ is a word at distance w from the code, or it is not. If it is not, the probability of $\mathbf{c}$ being a valid ciphertext is exactly 0. On the other hand, suppose $\mathbf{c}$ is at distance w from the code. Then there is only one choice of $\boldsymbol{\rho}$ with probability $1/q^{k-k'}$ and one choice of $\mathbf{e}$ with probability $1/\binom{n-k}{w-\mathsf{wt}(\boldsymbol{\mu})}$ that satisfy the equation (line 2 and line 4 in procedure $\mathsf{PKE}_2.\mathsf{Enc}$ in Figure 7), i.e. the probability of $\mathbf{c}$ being a valid PKE_2 ciphertext is exactly $\gamma = (1/q^{k-k'}) \cdot (1/\binom{n-k}{w-\mathsf{wt}(\boldsymbol{\mu})})$. Therefore, $\Pr_{\mathbf{r} \longleftarrow \mathcal{R}}[\mathbf{c} = \mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r})] \leq \gamma$ for any $\mathbf{c} \in (\mathsf{GF}(q))^{n-k}$ which completes the proof. ■ (of Lemma 1)
Now consider a query $\mathsf{CVO}(\mathbf{c})$. Let $\mathbf{m}' \longleftarrow \mathsf{PKE}_2.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathbf{c})$. If $\mathsf{CVO}(\mathbf{c}) \longrightarrow$

- The challenger $\mathcal{S}$ generates $\mathsf{param} \longleftarrow \mathsf{PKE}_1.\mathsf{Setup}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{PKE}_1.\mathsf{KeyGen}(\mathsf{param})$ for a security parameter λ and sends $\mathsf{param}, \mathsf{pk}$ to $\mathcal{A}$.
- The challenger $\mathcal{S}$ chooses a message $\mathbf{m}^* \in \mathcal{M}$, computes the challenge ciphertext $\mathsf{CT}^* = (\mathbf{c}^*, \mathbf{d}^*) \longleftarrow \mathsf{PKE}_1.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}^*; \mathbf{r}^*)$ and sends it to $\mathcal{A}$.
- The adversary $\mathcal{A}$ having access to the oracle O_{PCVA} i.e. the oracle $\mathsf{PCO}(\cdot,\cdot)$ and the oracle $\mathsf{CVO}(\cdot)$ (described in Figure 1), outputs $\mathbf{m}'$. Note that the oracle PCO takes a message $\mathbf{m}$ and a ciphertext CT as input and checks if the message recovered from CT is $\mathbf{m}$ or not while the oracle CVO takes a ciphertext CT as input distinct from the challenge ciphertext CT^* and checks whether the message recovered from CT belongs to the message space or not.

Fig. 8. Game $\mathbf{G}_0$ in the proof of the Theorem 4

- The challenger $\mathcal{S}$ generates $\mathsf{param} \longleftarrow \mathsf{PKE}_2.\mathsf{Setup}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \longleftarrow \mathsf{PKE}_2.\mathsf{KeyGen}(\mathsf{param})$ for a security parameter λ and sends $\mathsf{param}, \mathsf{pk}$ to $\mathcal{A}$.
- The challenger $\mathcal{S}$ chooses a message $\mathbf{m}^* \in \mathcal{M}$, computes $\mathbf{c}^* \longleftarrow \mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}^*; \mathbf{r}^*)$ and sends it to $\mathcal{A}$.
- The adversary $\mathcal{A}$ having access to the oracle O_{PCVA} (the oracle $\mathsf{PCO}(\cdot,\cdot)$ and the oracle $\mathsf{CVO}(\cdot)$) along with the hash oracle $\mathcal{G}(\cdot)$ (described in Figure 10), outputs $\mathbf{m}'$.

Fig. 9. Sequence of games $\mathbf{G}_j, j = 1, 2, 3, 4$ in the proof of the Theorem 4

1 in game $\mathbf{G}_2$, then $\exists\ (\mathbf{m}, \mathbf{r}) \in Q_{\mathcal{G}}$ with $\mathbf{m}' = \mathbf{m}$ and $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathbf{c}$ (line 11 in oracle CVO in Figure 10). This means that $\mathcal{G}(\mathbf{m}') = \mathcal{G}(\mathbf{m}) = \mathbf{r}$ and hence $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}'; \mathcal{G}(\mathbf{m}')) \longrightarrow \mathbf{c}$. This implies $\mathsf{CVO}(\mathbf{c}) \longrightarrow 1$ in game $\mathbf{G}_1$. If $\mathsf{CVO}(\mathbf{c}) \longrightarrow 1$ in game $\mathbf{G}_1$ then we can have $\mathsf{CVO}(\mathbf{c}) \longrightarrow 0$ in game $\mathbf{G}_2$ if $\mathcal{G}(\mathbf{m}')$ was not queried before in game $\mathbf{G}_2$. Let L be the event that $\mathcal{G}(\mathbf{m}')$ was not queried before the CVO oracle query. Games $\mathbf{G}_1$ and $\mathbf{G}_2$ are exactly the same unless L occurs. Suppose that $\mathbf{c}$ is a valid ciphertext with respect to param, pk i.e., there exists $\mathbf{m}$ and $\mathbf{r}$ such that $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathbf{c}$. Then the probability of the event L for a single query is $2^{-\gamma}$ where γ is the parameter defined in Lemma 1. On the contrary, if $\mathbf{c}$ is an invalid ciphertext with respect to param and pk, the event L does not occur. As the adversary $\mathcal{A}$ can make at most n_V queries to the oracle CVO, we obtain

$$|\Pr[E_1] - \Pr[E_2]| \leq n_V \cdot 2^{-\gamma}.$$

Game $\mathbf{G}_3$: In game $\mathbf{G}_3$, the oracles $\mathsf{PCO}(\mathbf{m}, \mathbf{c})$ and $\mathsf{CVO}(\mathbf{c})$ are simulated by the challenger $\mathcal{S}$ without checking $\mathbf{m} = \mathbf{m}'$ where $\mathbf{m}' \longleftarrow \mathsf{PKE}_2.\mathsf{Dec}(\mathsf{param}, \mathsf{sk}, \mathbf{c})$ (line 10 of PCO oracle and line 18 of CVO oracle in 10). Note that, in games $\mathbf{G}_2$ and $\mathbf{G}_3$, the adversary makes at most $n_{\mathcal{G}}$ distinct queries $\mathcal{G}(\boldsymbol{m_1}), \mathcal{G}(\boldsymbol{m_2}), \ldots,$ $\mathcal{G}(\boldsymbol{m_{n_\mathcal{G}}})$ to the hash oracle $\mathcal{G}$. We say such a query $\mathcal{G}(\boldsymbol{m_i})$ is problematic if and only if it exhibits a correctness error in the scheme PKE_2. As there is no correctness error in PKE_2, no query $\mathcal{G}(\boldsymbol{m_i})$ is problematic. Consequently, games $\mathbf{G}_2$ and $\mathbf{G}_3$ behave identically. Indeed, games $\mathbf{G}_2$ and $\mathbf{G}_3$ differ if the adversary $\mathcal{A}$ submits a PCO query on $(\mathbf{m}, \mathbf{c})$ or a CVO query on $\mathbf{c}$ together with a $\mathcal{G}$ query on $\mathbf{m}$ such that $\mathcal{G}(\mathbf{m})$ is problematic and $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathcal{G}(\mathbf{m})) \longrightarrow \mathbf{c}$. In this case, the challenger $\mathcal{S}$ will answer the query with 0 in game $\mathbf{G}_2$ as $\mathbf{m}' \neq \mathbf{m}$, whereas $\mathcal{S}$ will answer the query with 1 in game $\mathbf{G}_3$. Hence, we have

$$\Pr[E_3] = \Pr[E_2].$$

```
PCO(m, c)

1. for games $\mathbf{G}_1, \mathbf{G}_2$ do
2.   $\mathbf{m}' \longleftarrow \mathsf{PKE}_2.\mathsf{Dec}(\mathsf{sk}, \mathbf{c}, \mathsf{param})$;
3.   if $\mathbf{m}' = \mathbf{m}$ and
         $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}'; \mathcal{G}(\mathbf{m}')) \longrightarrow \mathbf{c}$
4.     return 1;
5.   else
6.     return 0;
7.   end if
8. end for
9. for games $\mathbf{G}_3, \mathbf{G}_4$ do
10.  if $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathcal{G}(\mathbf{m})) \longrightarrow \mathbf{c}$
11.    return 1;
12.  else
13.    return 0;
14.  end if
15. end for
```

```
G(m)

1. for game $\mathbf{G}_j, j = 1, 2, 3, 4$ do
2.   if $\exists$ $\mathbf{r}$ such that $(\mathbf{m}, \mathbf{r}) \in Q_{\mathcal{G}}$
3.     return $\mathbf{r}$;
4.   end if
5. end for
6. for game $\mathbf{G}_4$ do
7.   if $\mathbf{m} = \mathbf{m}^*$;
8.     QUERY = true;
9.     abort;
10.  end if
11. end for
12. for game $\mathbf{G}_j, j = 1, 2, 3, 4$ do
13.   $\mathbf{r} \overset{U}{\longleftarrow} R$;
14.   $Q_{\mathcal{G}} = Q_{\mathcal{G}} \cup \{(\mathbf{m}, \mathbf{r})\}$;
15.   return $\mathbf{r}$;
16. end for
```

```
CVO(c ≠ c*)

1. for game $\mathbf{G}_1$ do
2.   $\mathbf{m}' \longleftarrow \mathsf{PKE}_2.\mathsf{Dec}(\mathsf{sk}, \mathbf{c}, \mathsf{param})$;
3.   if $\mathbf{m}' \in \mathcal{M}$ and
         $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}'; \mathcal{G}(\mathbf{m}')) \longrightarrow \mathbf{c}$
4.     return 1;
5.   else
6.     return 0;
7.   end if
8. end for
9. for games $\mathbf{G}_2$ do
10.  $\mathbf{m}' \longleftarrow \mathsf{PKE}_2.\mathsf{Dec}(\mathsf{sk}, \mathbf{c}, \mathsf{param})$;
11.  if $\exists(\mathbf{m}, \mathbf{r}) \in Q_{\mathcal{G}}$ and $\mathbf{m}' = \mathbf{m}$ and
         $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathbf{c}$
12.    return 1;
13.  else
14.    return 0;
15.  end if
16. end for
17. for games $\mathbf{G}_3, \mathbf{G}_4$ do
18.   if $\exists(\mathbf{m}, \mathbf{r}) \in Q_{\mathcal{G}}$ and
          $\mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}; \mathbf{r}) \longrightarrow \mathbf{c}$
19.     return 1;
20.   else
21.     return 0;
22.   end if
23. end for
```

Fig. 10. The Plaintext Checking Oracle $\mathsf{PCO}(\cdot,\cdot)$, Ciphertext Validity Oracle $\mathsf{CVO}(\cdot,\cdot)$ and hash oracle $\mathcal{G}(\cdot)$ for games $\mathbf{G}_j, j = 1, 2, 3, 4$

Game $\mathbf{G}_4$: In game $\mathbf{G}_4$, the challenger $\mathcal{S}$ sets a flag QUERY=true and abort (with uniform random output), when the adversary $\mathcal{A}$ queries the hash oracle $\mathcal{G}$ on $\mathbf{m}^*$. Hence, games $\mathbf{G}_3$ and $\mathbf{G}_4$ differ if the flag QUERY=true is raised, meaning that $\mathcal{A}$ made a query $\mathcal{G}$ on $\mathbf{m}^*$, or, equivalently, $(\mathbf{m}^*, \cdot) \in Q_{\mathcal{G}}$. Thus, the games $\mathbf{G}_3$ and $\mathbf{G}_4$ are identical unless QUERY = true occurs. Therefore,

$$|\Pr[E_3] - \Pr[E_4]| \leq \Pr[\mathsf{QUERY} = \text{true}].$$

To bound $\Pr[E_4]$ we construct an adversary $\mathcal{A}'$ against the OW-CPA security of PKE_2 simulating game $\mathbf{G}_4$ for $\mathcal{A}$ in Figure 11. The adversary $\mathcal{A}'$ takes $\mathsf{param}, \mathsf{pk}, \mathbf{c}^*$ as input, perfectly simulates game $\mathbf{G}_4$ for $\mathcal{A}$ and finally outputs $\mathbf{m}' = \mathbf{m}^*$ if $\mathcal{A}$ wins in game $\mathbf{G}_4$. Here, $\mathcal{A}$ uses the same PCO, CVO oracles for game $\mathbf{G}_4$ in Figure 10 with the same hash oracle $\mathcal{G}$ for game $\mathbf{G}_3$ in Figure 10. Hence,

$$\Pr[E_4] = \mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}_2}(\mathcal{A}').$$

```
A'(param, pk, c*)

  1. $\mathbf{m}' \longleftarrow \mathcal{A}^{\mathcal{G}(\cdot),\mathsf{PCO}(\cdot,\cdot),\mathsf{CVO}(\cdot)}(\mathsf{param}, \mathsf{pk}, \mathbf{c}^*)$;
  2. return $\mathbf{m}'$;
```

Fig. 11. Adversary $\mathcal{A}'$ against OW-CPA security of PKE_2

Combining all the probabilities, we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{OW\text{-}PCVA}}_{\mathsf{PKE}_1}(\mathcal{A}) = |\Pr[E_0]| &= |\Pr[E_1]| \\
&= |\Pr[E_1] - \Pr[E_2] + \Pr[E_2]| \\
&\le |\Pr[E_1] - \Pr[E_2]| + |\Pr[E_2]| \\
&\le n_V \cdot 2^{-\gamma} + |\Pr[E_3]| \\
&= n_V \cdot 2^{-\gamma} + |\Pr[E_3] - \Pr[E_4] + \Pr[E_4]| \\
&\le n_V \cdot 2^{-\gamma} + |\Pr[E_3] - \Pr[E_4]| + |\Pr[E_4]| \\
&\le n_V \cdot 2^{-\gamma} + \Pr[\mathsf{QUERY} = \mathsf{true}] + \mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}_2}(\mathcal{A}')
\end{aligned}$$

We consider the following relation between OW-CPA security and IND-CPA security of a public key encryption scheme (Remark 1)

Lemma 2. *[35] Let PKE be a public key encryption scheme. Then, for any OW-CPA adversary $\mathcal{B}$, there exists an IND-CPA adversary $\mathcal{A}$ with the same running time as that of $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathsf{OW\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{B}) \le \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}}(\mathcal{A}) + 1/|\mathcal{M}|$$

where $\mathcal{M}$ is the message space.

As $\mathcal{M} = (\mathsf{GF}(q))^{k'}$ in PKE_2, we have by Lemma 2, we have

$$\mathsf{Adv}^{\mathsf{OW\text{-}PCVA}}_{\mathsf{PKE}_1}(\mathcal{A}) \le n_V \cdot 2^{-\gamma} + \Pr[\mathsf{QUERY} = \mathsf{true}] + \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}_2}(\mathcal{A}'') + \frac{1}{q^{k'}}$$

for an IND-CPA adversary $\mathcal{A}''$. We now construct another adversary $\mathcal{D}$ (Figure 12) against the IND-CPA security of the PKE_2 which wins when the flag QUERY=true is set in game $\mathbf{G}_4$. The adversary $\mathcal{D}$ selects two random messages $\mathbf{m}_0^*, \mathbf{m}_1^*$ and runs $\mathcal{A}$ on $(\mathsf{param}, \mathsf{pk}, \mathbf{c}^*)$ where $\mathbf{c}^* \longleftarrow \mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}_b^*; \mathbf{r}_b^*)$, $b \xleftarrow{U} \{0,1\}$, is generated and sent by the IND-CPA challenger $\mathcal{C}_h$ in the IND-CPA security game between $\mathcal{C}_h$ and $\mathcal{D}$. Now consider the IND-CPA security game for the adversary $\mathcal{D}$ with random challenge bit b. Let Z be the event that $\mathcal{A}$ queries random oracle $\mathcal{G}$ on $\mathbf{m}_{1-b}^*$. Since the message $\mathbf{m}_{1-b}^*$ is taken uniformly from $\mathcal{M}$ and independent from $\mathcal{A}$'s view, we have $\Pr[Z] \le \frac{n_{\mathcal{G}}}{q^{k'}}$. Let us now assume the

```
$\underline{\mathcal{D}(\mathsf{param}, \mathsf{pk})}$

1. $(\mathbf{m}_0^*, \mathbf{m}_1^*) \xleftarrow{U} \mathcal{M} \times \mathcal{M}$;
2. $\mathbf{m}' \longleftarrow \mathcal{A}^{\mathcal{G}(\cdot),\mathsf{PCO}(\cdot,\cdot),\mathsf{CVO}(\cdot)}(\mathsf{param}, \mathsf{pk}, \mathbf{c}^* \longleftarrow \mathsf{PKE}_2.\mathsf{Enc}(\mathsf{param}, \mathsf{pk}, \mathbf{m}_b^*; \mathbf{r}_b^*))$;
3. $b' = \begin{cases} 0 & \text{for } |Q_{\mathcal{G}}(\mathbf{m}_0^*)| = 1, |Q_{\mathcal{G}}(\mathbf{m}_1^*)| = 0; \\ 1 & \text{for } |Q_{\mathcal{G}}(\mathbf{m}_0^*)| = 0, |Q_{\mathcal{G}}(\mathbf{m}_1^*)| = 1; \\ \xleftarrow{U} \{0,1\} & \text{for } |Q_{\mathcal{G}}(\mathbf{m}_0^*)| = |Q_{\mathcal{G}}(\mathbf{m}_1^*)|; \end{cases}$
4. return $b'$;
```

Fig. 12. Adversary $\mathcal{D}$ against IND-CPA security of PKE_2

event that Z did not happen which leads $|Q_{\mathcal{G}}(\mathbf{m}_{1-b}^*)| = 0$. Here $|Q_{\mathcal{G}}(\mathbf{m})|$ denotes the number of all $(\mathbf{m}, \mathbf{r}) \in Q_{\mathcal{G}}$ for a fixed $\mathbf{m} \in \mathcal{M}$. Note that, $|Q_{\mathcal{G}}(\mathbf{m})|$ is either 1 or 0 for a message $\mathbf{m}$. When QUERY=true occurs, the adversary $\mathcal{A}$ queried the random oracle $\mathcal{G}$ on $\mathbf{m}_b^*$. Hence, $|Q_{\mathcal{G}}(\mathbf{m}_b^*)| = 1, |Q_{\mathcal{G}}(\mathbf{m}_{1-b}^*)| = 0$ (as Z did not happen) and therefore $b = b'$. When QUERY=true does not happen, $\mathcal{A}$ did not query the oracle $\mathcal{G}$ on $\mathbf{m}_b^*$. So, $|Q_{\mathcal{G}}(\mathbf{m}_b^*)| = Q_{\mathcal{G}}(\mathbf{m}_{1-b}^*)| = 0$ and $\Pr[b = b'] = 1/2$ as $\mathcal{D}$ selects a random bit b'. From all of these relations we can write

$$\begin{aligned} |\Pr[b = b'|\overline{Z}] - 1/2| + |\Pr[b = b'|Z]| &\geq |\Pr[b = b'|Z] + \Pr[b = b'|\overline{Z}] - 1/2| \\ &\geq |\Pr[b = b'|\overline{Z}] - 1/2| \end{aligned}$$

which yields

$$\begin{aligned} \mathsf{Adv}_{\mathsf{PKE}_2}^{\mathsf{IND\text{-}CPA}}(\mathcal{D}) + \frac{n_{\mathcal{G}}}{q^{k'}} &\geq |\Pr[b = b'|\overline{Z}] - 1/2| \\ &= |\Pr[b = b'|\mathsf{QUERY} = \mathsf{true}] \cdot \Pr[\mathsf{QUERY} = \mathsf{true}] \\ &+ \Pr[b = b'|\overline{\mathsf{QUERY} = \mathsf{true}}] \cdot \Pr[\overline{\mathsf{QUERY} = \mathsf{true}}] - 1/2| \\ &= |\Pr[\mathsf{QUERY} = \mathsf{true}] + 1/2\Pr[\overline{\mathsf{QUERY} = \mathsf{true}}] - 1/2| \\ &= |\Pr[\mathsf{QUERY} = \mathsf{true}] + 1/2(1 - \Pr[\mathsf{QUERY} = \mathsf{true}]) - 1/2| \\ &= 1/2\Pr[\mathsf{QUERY} = \mathsf{true}]. \end{aligned}$$

Using the above relation and combining two IND-CPA adversaries $\mathcal{A}''$ and $\mathcal{D}$ to a new IND-CPA adversary $\mathcal{B}$ we get

$$\mathsf{Adv}_{\mathsf{PKE}_1}^{\mathsf{OW\text{-}PCVA}}(\mathcal{A}) \leq n_V \cdot 2^{-\gamma} + (2n_{\mathcal{G}} + 1)/q^{k'} + 3 \cdot \mathsf{Adv}_{\mathsf{PKE}_2}^{\mathsf{IND\text{-}CPA}}(\mathcal{B})$$

which completes the proof. ■

The OW-PCVA security for a PKE scheme (Definition 4 described in Subsection 2.1) trivially implies the OW-VA security of a PKE scheme considering zero queries to the $\mathsf{PCO}(\cdot,\cdot)$ oracle (see Remark 2). Therefore, the following corollary is an immediate consequence of Theorem 4.

Corollary 1. *If the scheme* PKE_2 =(Setup, KeyGen, Enc, Dec) *described in Figure 7 is* IND-CPA *secure (Definition 3 in Subsection 2.1), then the public key encryption scheme* $\mathsf{PKE}_1 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *as described in Figure 2* *provides* OW-VA *security (Definition 4 in Subsection 2.1) when the hash function* $\mathcal{G}$ *is modeled as a random oracle.*

Theorem 5. *If the decisional* SD *problem (Definition 8 in Subsection 2.3) is hard, the public key matrix* H *(derived from the public key* pk *which is generated by running* $\mathsf{PKE}_2.\mathsf{KeyGen}(\mathsf{param})$ *where* $\mathsf{param} \longleftarrow \mathsf{PKE}_2.\mathsf{Setup}(1^\lambda)$*) is indistinguishable (Definition 9 in Subsection 2.3) and the hash function* $\mathcal{G}$ *is modeled as a random oracle, then the public key encryption scheme* $\mathsf{PKE}_2 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *presented in Figure 7 is* IND-CPA *secure (Definition 3 in Subsection 2.1).*

Proof. In the scheme PKE_2, a ciphertext $\mathbf{c}$ is computed using the $(n-k)\times n$ public key matrix $H = [M|I_{n-k}]$ which is generated from the public key pk=$\{\boldsymbol{\psi}_{i,j} \mid i = 0, 1, \ldots, mt-1,\ j = 0, 1, \ldots, \frac{k}{s} - 1\}$ by setting $(n-k)\times n$ matrix

$$M = \begin{bmatrix} M_{0,0} & M_{0,1} & \cdots & M_{0,\frac{k}{s}-1} \\ M_{1,0} & M_{1,1} & \cdots & M_{1,\frac{k}{s}-1} \\ \cdots & \cdots & \cdots & \cdots \\ M_{mt-1,0} & M_{mt-1,1} & \cdots & M_{mt-1,\frac{k}{s}-1} \end{bmatrix}$$

where $M_{i,j}$ is a dyadic matrix of order s with signature $\boldsymbol{\psi}_{i,j} \in (\mathsf{GF}(q))^s$ and $n - k = mst$ (see the procedure $\mathsf{PKE}_2.\mathsf{Enc}$ in Figure 7).
We compute the ciphertext as $\mathbf{c} = H(\mathbf{e}')^T$ where $\mathbf{e}' = (\mathbf{e}||\boldsymbol{\mu}) = (\mathbf{e}||\boldsymbol{\rho}||\mathbf{m}) = (\mathbf{r}_1||\mathbf{m})$, where $\mathbf{r}_1 = (\mathbf{e}||\boldsymbol{\rho}) \in (\mathsf{GF}(q))^{n-k'}, \boldsymbol{\rho} \in (\mathsf{GF}(q))^{k-k'}$ satisfying $\mathbf{r} = \mathcal{G}(\mathbf{m}) = (\boldsymbol{\rho}||\boldsymbol{\sigma})$ with $\boldsymbol{\sigma} \in (\mathsf{GF}(q))^{k'}$. Note that $\boldsymbol{\sigma}$ is used as a seed to generate the error vector $\mathbf{e} \in (\mathsf{GF}(q))^{n-k}$ with $\mathsf{wt}(\mathbf{e}) = w - \mathsf{wt}(\boldsymbol{\mu})$ (see Figure 7). Let $H = [H_1|H_2]$ where H_1 and H_2 are respectively $(n-k)\times(n-k')$ and $(n-k)\times k'$ sub-matrices of H. Therefore,

$$\mathbf{c} = H[\mathbf{r}_1||\mathbf{m}]^T = H_1\mathbf{r}_1^T + H_2\mathbf{m}^T.$$

Let us assume that there exists a PPT algorithm $\mathcal{D}$ such that

$$\begin{aligned} |\Pr[\mathcal{D}(H, H_1\mathbf{r}_1^T) = 1 \mid \mathbf{r}_1 \xleftarrow{U} (\mathsf{GF}(q))^{n-k'}, H = [H_1|H_2]] \\ - \Pr[\mathcal{D}(H, \mathbf{s}) = 1 \mid \mathbf{s} \xleftarrow{U} U_{(n-k)\times 1}, H = [H_1|H_2]]| \geq \delta \end{aligned}$$

for a positive small δ where $U_{c\times d}$ is the uniform distribution over $c \times d$ random q-ary matrices. Let succ be the event that $\mathcal{D}(H, H_1\mathbf{r}_1^T) = 1$ where $\mathbf{r}_1 \xleftarrow{U} (\mathsf{GF}(q))^{n-k'}$ and H is the public key matrix. We construct an adversary $\mathcal{D}'$ (as in Figure 13) which distinguishes a random matrix from H. Let $\mathsf{E}_{\mathsf{rand}}$ be the event that the matrix R was chosen randomly from uniform distribution $U_{(n-k)\times n}$ and $\mathsf{E}_{\mathsf{real}}$ be the event that R be the public key matrix H constructed as stated above

```
D'(R)

1. $R_{(n-k)\times n} = [(R_1)_{(n-k)\times(n-k')}||(R_2)_{(n-k)\times k'}]$;
2. $p \xleftarrow{U} \{0,1\}$;
3. if $p = 1$
4.     $\mathbf{s}_1 = R_1\mathbf{r}_1^T$;
5.     $p' \longleftarrow \mathcal{D}(R, \mathbf{s}_1)$;
6. else
7.     $\mathbf{s}_0 \xleftarrow{U} U_{n-k\times 1}$;
8.     $p' \longleftarrow \mathcal{D}(R, \mathbf{s}_0)$;
9. end if
10. if $p = p'$
11.     return 1;
12. else
13.     return 0;
14. end if
```

Fig. 13. Adversary $\mathcal{D}'$ in the proof of Theorem 5

using the public key pk generated $\mathsf{PKE}_2.\mathsf{KeyGen}(\mathsf{param})$. Then

$$\begin{aligned}&|\Pr[p = p'|\mathsf{E}_{\mathsf{real}}] - \Pr[p = p'|\mathsf{E}_{\mathsf{rand}}|\\ =&\Pr[\mathcal{D}'(H) = 1|\text{public key matrix } H = [M|I_{n-k}] \in (\mathsf{GF}(q))^{(n-k)\times n}]\\ &\qquad\qquad\qquad\qquad - \Pr[\mathcal{D}'(R) = 1|R \xleftarrow{U} U_{(n-k)\times n}]\\ =&\mathsf{Adv}^{\mathsf{IND}}_{\mathcal{D}',H}(\lambda) \leq \mathsf{Adv}^{\mathsf{IND}}_{H}(\lambda)\end{aligned}$$

When $\mathsf{E}_{\mathsf{real}}$ occurs, we have $R = H$ which is distributed exactly as in the real execution. Since $\mathcal{D}'$ outputs 1 if and only if $\mathcal{D}$ succeeds, we have $\Pr[p = p'|\mathsf{E}_{\mathsf{real}}] = \Pr[\mathsf{succ}]$. When $\mathsf{E}_{\mathsf{rand}}$ occurs, the matrix R was chosen randomly from uniform distribution $U_{(n-k)\times n}$. Therefore,

$$\begin{aligned}&|\Pr[p = p'|\mathsf{E}_{\mathsf{rand}}] - 1/2|\\ =&|\Pr[\mathcal{D}'(R) = 1|R \xleftarrow{U} U_{(n-k)\times n}] - 1/2|\\ =&|\Pr[\mathcal{D}(R, \mathbf{s}_1) = 1|R \xleftarrow{U} U_{(n-k)\times n}] - \Pr[\mathcal{D}(R, \mathbf{s}_0) = 1|R \xleftarrow{U} U_{(n-k)\times n}|\\ =&\mathsf{Adv}^{\mathsf{DEC}}_{\mathcal{D},\mathsf{SD}} \leq \mathsf{Adv}^{\mathsf{DEC}}_{\mathsf{SD}}(\lambda)\end{aligned}$$

where $\Pr[\mathcal{D}(R, \mathbf{s}_0) = 1|R \xleftarrow{U} U_{(n-k)\times n}] = 1/2$. Combining all the probabilities together, we get

$$\begin{aligned}\delta \leq\ &|\Pr[\mathcal{D}(H, H_1\mathbf{r}_1^T) = 1 \mid \mathbf{r}_1 \xleftarrow{U} (\mathsf{GF}(q))^{n-k'}, H = [H_1|H_2]]\\ &\qquad - \Pr[\mathcal{D}(H, \mathbf{s}) = 1 \mid \mathbf{s} \xleftarrow{U} U_{(n-k)\times 1}, H = [H_1|H_2]]|\\ =\ &|\Pr[\mathsf{succ}] - 1/2|\\ =\ &|\Pr[p = p'|\mathsf{E}_{\mathsf{real}}] - 1/2|\\ =\ &|\Pr[p = p'|\mathsf{E}_{\mathsf{real}}] - \Pr[p = p'|\mathsf{E}_{\mathsf{rand}} + \Pr[p = p'|\mathsf{E}_{\mathsf{rand}} - 1/2|\\ \leq\ &\mathsf{Adv}^{\mathsf{IND}}_{H}(\lambda) + |\Pr[p = p'|\mathsf{E}_{\mathsf{rand}}] - 1/2|\\ \leq\ &\mathsf{Adv}^{\mathsf{DEC}}_{\mathsf{SD}}(\lambda) + \mathsf{Adv}^{\mathsf{IND}}_{H}(\lambda).\end{aligned}$$

Note that, when the algorithm $\mathcal{D}$ takes uniformly distributed inputs, it outputs 1 with probability 1/2, i.e. $\Pr[\mathcal{D}(H, \mathbf{s}) = 1 \mid \mathbf{s} \xleftarrow{U} U_{(n-k)\times 1}, H = [H_1|H_2]] = 1/2$. Therefore, we get the relation $\delta \leq \mathsf{Adv}^{\mathsf{OW}}_{\mathsf{SD}}(\lambda) + \mathsf{Adv}^{\mathsf{IND}}_{H}(\lambda)$. This is a contradiction since decisional SD problem is hard and H is indistinguishable. Hence,

$$\begin{aligned}|\Pr[\mathcal{D}(H, H_1\mathbf{r}_1^T) = 1 \mid \mathbf{r}_1 \xleftarrow{U} (\mathsf{GF}(q))^{n-k'}, H = [H_1|H_2]] \\ - \Pr[\mathcal{D}(H, \mathbf{s}) = 1 \mid \mathbf{s} \xleftarrow{U} U_{(n-k)\times 1}, H = [H_1|H_2]]| \leq \delta\end{aligned} \tag{1}$$

Now, we construct a distinguisher $\mathcal{B}$ from an IND-CPA adversary $\mathcal{A}$ against the scheme PKE_2 as in Figure 14 where $\mathcal{B}$ distinguishes $\tilde{\mathbf{s}}_1 = H_1\mathbf{r}_1^T$ from the same length random value $\tilde{\mathbf{s}}_0$ where $\mathbf{r}_1 \xleftarrow{U} (\mathsf{GF}(q))^{n-k'}$. In Figure 14, H_2 is extracted

```
B(param, pk, s̃)

1. $(\mathbf{m}_0, \mathbf{m}_1) \longleftarrow \mathcal{A}(\mathsf{param}, \mathsf{pk})$;
2. $b \xleftarrow{U} \{0, 1\}$;
3. $\mathbf{c} = \tilde{\mathbf{s}} + H_2\mathbf{m}_b^T$;
4. $b' \longleftarrow \mathcal{A}(\mathbf{c})$;
5. if $b = b'$
6.     return 1;
7. else
8.     return 0;
9. end if
```

Fig. 14. A distinguisher $\mathcal{B}$ from the IND-CPA adversary $\mathcal{A}$ in the proof of Theorem 5

by $\mathcal{B}$ from $H = [H_1|H_2]$ which is derived from pk. Let rand be the event that $\tilde{\mathbf{s}}(= \tilde{\mathbf{s}}_0)$ was chosen from the uniform distribution $U_{(n-k)\times 1}$ while real be the event that $\tilde{\mathbf{s}}(= \tilde{\mathbf{s}}_1)$ is $H_1\mathbf{r}_1^T$. We will say that $\mathcal{A}$ succeeds if $b = b'$ under the event real occurs. We denote this event win. Note that

$$|\Pr[\mathcal{B}(\mathsf{param}, \mathsf{pk}, \tilde{\mathbf{s}}) = 1|\mathsf{real}] - \Pr[\mathcal{B}(\mathsf{param}, \mathsf{pk}, \tilde{\mathbf{s}}) = 1|\mathsf{rand}]| \leq \mathsf{Adv}^{\mathsf{IND}}_{H_1\mathbf{r}_1^T}(\lambda). \tag{2}$$

Note that when event real occurs we have $\tilde{\mathbf{s}} = \tilde{\mathbf{s}}_1 = H_1\mathbf{r}_1^T$ which is distributed exactly as in the real execution as $\mathbf{c} = \tilde{\mathbf{s}} + H_2\mathbf{m}_b^T = H_1\mathbf{r}_1^T + H_2\mathbf{m}_b^T = H[\mathbf{r}_1||\mathbf{m}_b]^T$. Since $\mathcal{B}$ outputs 1 if and only if $\mathcal{A}$ succeeds, we have $\Pr[\mathcal{B}(\mathsf{param}, \mathsf{pk}, \tilde{\mathbf{s}}) = 1|\mathsf{real}] = \Pr[\mathsf{win}]$. On the other hand, when event rand occurs, $\tilde{\mathbf{s}}$ is distributed uniformly. Therefore, $\tilde{\mathbf{s}} + H_2\mathbf{m}_b^T$ given to $\mathcal{A}$ is uniformly distributed as well. This means that $\mathcal{A}$ obtains no information related to b. Since $\mathcal{B}$ outputs 1 if and only if $\mathcal{A}$ succeeds, we can conclude that $\Pr[\mathcal{B}(\mathsf{param}, \mathsf{pk}, \tilde{\mathbf{s}}) = 1|\mathsf{rand}] = 1/2$. By combining these results, we get

$$\begin{aligned}|\Pr[\mathcal{B}(\mathsf{param}, \mathsf{pk}, \tilde{\mathbf{s}}) = 1|\mathsf{real}] - \Pr[\mathcal{B}(\mathsf{param}, \mathsf{pk}, \tilde{\mathbf{s}}) = 1|\mathsf{rand}]| &= |\Pr[\mathsf{win}] - 1/2| \\ &= \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}_2}(\mathcal{A}).\end{aligned} \tag{3}$$

From equation (2) and (3), we can say that if $\mathcal{A}$ breaks the IND-CPA security with non-negligible probability, the distinguisher $\mathcal{B}$ distinguishes $\tilde{\mathbf{s}}_1 = H_1 r_1^T$ from the same length random value $\tilde{\mathbf{s}}_0$ with non-negligible probability. More precisely, if $\mathsf{Adv}^{\mathsf{IND}}_{H_1\mathbf{r}_1^T}(\lambda) \leq \delta$, then $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE}_2}(\mathcal{A}) \leq \delta$. From Equation (1), we can conclude that PKE_2 is IND-CPA secure provided decisional SD problem is hard and H is indistinguishable. ■

Theorem 6. *Assuming the hardness of decisional* SD *problem (Definition 8 in Subsection 2.3) and indistinguishability of the public key matrix* H *(derived from the public key* pk *by running* $\mathsf{KEM.KeyGen(param)}$ *where* $\mathsf{param} \longleftarrow \mathsf{KEM.Setup}(1^\lambda)$*,* λ *being the security parameter), our* $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *described in Section 3 provides* $\mathsf{IND\text{-}CCA}$ *security (Definition 6 in Subsection 2.2) when the hash functions* $\mathcal{H}'$ *and* $\mathcal{G}$ *are modeled as random oracles.*

Proof. The proof of the above theorem is the immediate consequence of Theorem 3, Corollary 1 and Theorem 5.

Remark 3. For proving security in the quantum-accessible random oracle model, it is required to show post-quantum security of a construction where the adversary is able to query the random oracle with quantum access. We consider the security games in the quantum random oracle model (QROM) along with the classical random oracle model. In fact, the adversaries equipped with a quantum computer are provided quantum access to the random oracles and classical access to some other oracles like plaintext checking oracles, decapsulation oracles etc. The KEM protocol also provides security in quantum random oracle model by the Theorem 7 which follows the work in [35]. We can construct a public key encryption scheme PKE_1 and then prove that $\mathsf{OW\text{-}PCA}$ (One-Wayness under Plaintext Checking Attacks) security of the public key encryption scheme PKE_1 indicates the IND-CCA security of the KEM considering $\mathcal{H}, \mathcal{H}'$ as quantum random oracles (see Theorem 9). Then, we can form another public key encryption scheme PKE_2 and show that $\mathsf{OW\text{-}CPA}$ (One-Wayness under Chosen Plaintext Attacks) security of the encryption scheme PKE_2 implies $\mathsf{OW\text{-}PCA}$ security of the PKE_1 modeling $\mathcal{G}$ as a quantum random oracle (see Theorem 8). The scheme PKE_2 is $\mathsf{IND\text{-}CPA}$ secure as the syndrome decoding problem is hard and the public key matrix is indistinguishable (see Theorem 5). Combining $\mathsf{IND\text{-}CPA}$ security of the scheme PKE_2 and the fact that $\mathsf{IND\text{-}CPA}$ security always implies $\mathsf{OW\text{-}CPA}$ security, we can get Theorem 7.

Theorem 7. *Assuming the hardness of decisional* SD *problem (Definition 8 in Subsection 2.3) and indistinguishability of the public key matrix* H *(derived from the public key* pk *by running* $\mathsf{KEM.KeyGen(param)}$ *where* $\mathsf{param} \longleftarrow \mathsf{KEM.Setup}(1^\lambda)$*,* λ *being the security parameter), our* $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *described in Section 3 provides* $\mathsf{IND\text{-}CCA}$ *security (Definition 6 in Subsection 2.2) when the hash functions* $\mathcal{G}, \mathcal{H}$ *and* $\mathcal{H}'$ *are modeled as quantum random oracles.*

Note that, proof of Theorem 7 follows from Theorem 5, Remark 1, Theorem 8 and Theorem 9.

Theorem 8. *If the public key encryption scheme* PKE_2 =(Setup, KeyGen, Enc, Dec) *described in Figure 7 is* $\mathsf{OW\text{-}CPA}$ *secure (Definition 4 in Subsection 2.1), then the public key encryption scheme* $\mathsf{PKE}_1 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *as described in Figure 2 provides* $\mathsf{OW\text{-}PCA}$ *security (Definition 4 in Subsection 2.1) when the hash function* $\mathcal{G}$ *is modeled as a quantum random oracle.*

Theorem 9. *If the public key encryption scheme* $\mathsf{PKE}_1 = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *described in Figure 2 is* $\mathsf{OW\text{-}PCA}$ *secure (Definition 4 in Subsection 2.1) and there exist cryptographically secure hash functions, then the key encapsulation mechanism* $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *as described in Section 3 achieves* $\mathsf{IND\text{-}CCA}$ *security (Definition 6 in Subsection 2.2) when the hash function* $\mathcal{H}$ *and* $\mathcal{H}'$ *are modeled as quantum random oracles.*

5 Conclusion

In this work, we give a proposal to design an IND-CCA secure key encapsulation mechanism based on Generalized Srivastava codes. In terms of storage, our work seems well as compared to some other code-based KEM protocols as shown in Table 1. The scheme instantiated with Generalized Srivastava code does not involve any correctness error like some lattice-based schemes which allows achieving a simpler and tighter security bound for the IND-CCA security. In the upcoming days, it would be desirable to devise more efficient and secure constructions using suitable error-correcting codes.

References

1. Aguilar-Melchor, C., Blazy, O., Deneuville, J.C., Gaborit, P., Zémor, G.: Efficient encryption from random quasi-cyclic codes. IEEE Transactions on Information Theory 64(5), 3927–3943 (2018)
2. Albrecht, M., Cid, C., Paterson, K.G., Tjhai, C.J., Tomlinson, M.: Nts-kem. NIST submissions (2019)
3. Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Gueron, S., Guneysu, T., Melchor, C.A., et al.: Bike: Bit flipping key encapsulation. NIST submissions (2017)
4. Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Gueron, S., Guneysu, T., Melchor, C.A., et al.: Bike: Bit flipping key encapsulation. NIST submissions (2019)
5. Aragon, N., Blazy, O., Deneuville, J.C., Gaborit, P., Hauteville, A., Ruatta, O., Tillich, J.P., Zémor, G.: Lake-low rank parity check codes key exchange (2017)
6. Aragon, N., Blazy, O., Deneuville, J.C., Gaborit, P., Hauteville, A., Ruatta, O., Tillich, J.P., Zémor, G.: Locker-low rank parity check codes encryption (2017)
7. Baldi, M., Barenghi, A., Chiaraluce, F., Pelosi, G., Santini, P.: Ledakem: a post-quantum key encapsulation mechanism based on qc-ldpc codes. In: International Conference on Post-Quantum Cryptography. pp. 3–24. Springer (2018)
8. Baldi, M., Bodrato, M., Chiaraluce, F.: A new analysis of the mceliece cryptosystem based on qc-ldpc codes. In: International Conference on Security and Cryptography for Networks. pp. 246–262. Springer (2008)
9. Baldi, M., Chiaraluce, F.: Cryptanalysis of a new instance of mceliece cryptosystem based on qc-ldpc codes. In: 2007 IEEE International Symposium on Information Theory. pp. 2591–2595. IEEE (2007)
10. Baldi, M., Chiaraluce, F., Garello, R.: On the usage of quasi-cyclic low-density parity-check codes in the mceliece cryptosystem. In: 2006 First International Conference on Communications and Electronics. pp. 305–310. IEEE (2006)
11. Baldi, M., Chiaraluce, F., Garello, R., Mininni, F.: Quasi-cyclic low-density parity-check codes in the mceliece cryptosystem. In: 2007 IEEE International Conference on Communications. pp. 951–956. IEEE (2007)
12. Banegas, G., Barreto, P.S., Boidje, B.O., Cayrel, P.L., Dione, G.N., Gaj, K., Gueye, C.T., Haeussler, R., Klamti, J.B., N'diaye, O., et al.: Dags: Key encapsulation using dyadic gs codes. Journal of Mathematical Cryptology 12(4), 221–239 (2018)
13. Banegas, G., Barreto, P., Boidje, B.O., Cayrel, P.L., Dione, G.N., Gaj, K., Gueye, C.T., Haeussler, R., Klamti, J.B., N'diaye, O., et al.: Dags: Key encapsulation using dyadic gs codes. IACR Cryptology ePrint Archive 2017(1037) (2017)
14. Bardet, M., Barelli, E., Blazy, O., Canto-Torres, R., Couvreur, A., Gaborit, P., Otmani, A., Sendrier, N., Tillich, J.P.: Big quake. NIST submissions (2017)
15. Barelli, E., Couvreur, A.: An efficient structural attack on nist submission dags. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 93–118. Springer (2018)
16. Barg, A.: Complexity issues in coding theory. Electronic Colloquium on Computational Complexity (ECCC) 4(46) (1997)
17. Barreto, P.S., Cayrel, P.L., Misoczki, R., Niebuhr, R.: Quasi-dyadic cfs signatures. In: International Conference on Information Security and Cryptology. pp. 336–349. Springer (2010)
18. Barreto, P.S., Gueron, S., Gueneysu, T., Misoczki, R., Persichetti, E., Sendrier, N., Tillich, J.P.: Cake: Code-based algorithm for key encapsulation. In: IMA International Conference on Cryptography and Coding. pp. 207–226. Springer (2017)

19. Berlekamp, E., McEliece, R., Van Tilborg, H.: On the inherent intractability of certain coding problems (corresp.). IEEE Transactions on Information Theory 24(3), 384–386 (1978)
20. Bernstein, D.J., Chou, T., Lange, T., von Maurich, I., Misoczki, R., Niederhagen, R., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., et al.: Classic mceliece: conservative code-based cryptography. NIST submissions (2017)
21. Bernstein, D.J., Chou, T., Schwabe, P.: Mcbits: fast constant-time code-based cryptography. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 250–272. Springer (2013)
22. Cayrel, G.N.D., Gaj, K., Gueye, C.T., Haeussler, R., Klamti, J.B., N'diaye, O., Nguyen, D.T., Persichetti, E., Ricardini, J.E.: Dags: Reloaded revisiting dyadic key encapsulation
23. Cayrel, P.L., Hoffmann, G., Persichetti, E.: Efficient implementation of a cca2-secure variant of mceliece using generalized srivastava codes. In: International Workshop on Public Key Cryptography. pp. 138–155. Springer (2012)
24. Deneuville, J.C., Gaborit, P., Zémor, G.: Ouroboros: A simple, secure and efficient key exchange protocol based on coding theory. In: International Workshop on Post-Quantum Cryptography. pp. 18–34. Springer (2017)
25. Eaton, E., Lequesne, M., Parent, A., Sendrier, N.: Qc-mdpc: a timing attack and a cca2 kem. In: International Conference on Post-Quantum Cryptography. pp. 47–76. Springer (2018)
26. Fabšič, T., Hromada, V., Stankovski, P., Zajac, P., Guo, Q., Johansson, T.: A reaction attack on the qc-ldpc mceliece cryptosystem. In: International Workshop on Post-Quantum Cryptography. pp. 51–68. Springer (2017)
27. Faugere, J.C., Gauthier-Umana, V., Otmani, A., Perret, L., Tillich, J.P.: A distinguisher for high-rate mceliece cryptosystems. IEEE Transactions on Information Theory 59(10), 6830–6844 (2013)
28. Faugere, J.C., Otmani, A., Perret, L., De Portzamparc, F., Tillich, J.P.: Folding alternant and goppa codes with non-trivial automorphism groups. IEEE Transactions on Information Theory 62(1), 184–198 (2015)
29. Faugere, J.C., Otmani, A., Perret, L., De Portzamparc, F., Tillich, J.P.: Structural cryptanalysis of mceliece schemes with compact keys. Designs, Codes and Cryptography 79(1), 87–112 (2016)
30. Faugere, J.C., Otmani, A., Perret, L., Tillich, J.P.: Algebraic cryptanalysis of mceliece variants with compact keys. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 279–298. Springer (2010)
31. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Annual International Cryptology Conference. pp. 537–554. Springer (1999)
32. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of cryptology 26(1), 80–101 (2013)
33. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of computer and system sciences 28(2), 270–299 (1984)
34. Guo, Q., Johansson, T., Stankovski, P.: A key recovery attack on mdpc with cca security using decoding errors. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 789–815. Springer (2016)
35. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the fujisaki-okamoto transformation. In: Theory of Cryptography Conference. pp. 341–371. Springer (2017)

36. Kim, J.L., Kim, Y.S., Galvez, L., Kim, M.J., Lee, N.: Mcnie: A code-based public-key cryptosystem. arXiv preprint arXiv:1812.05008 (2018)
37. Li, Y.X., Deng, R.H., Wang, X.M.: On the equivalence of mceliece's and niederreiter's public-key cryptosystems. IEEE Transactions on Information Theory 40(1), 271–273 (1994)
38. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes, vol. 16. Elsevier (1977)
39. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. Coding Thv 4244, 114–116 (1978)
40. Melchor, C.A., Aragon, N., Bardet, M., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C.: Rollo-rank-ouroboros, lake & locker (2019)
41. Melchor, C.A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Hauteville, A., Zémor, G., Bourges, I.C.: Ouroboros-r (2017)
42. Misoczki, R., Barreto, P.S.: Compact mceliece keys from goppa codes. In: International Workshop on Selected Areas in Cryptography. pp. 376–392. Springer (2009)
43. Misoczki, R., Tillich, J.P., Sendrier, N., Barreto, P.S.: Mdpc-mceliece: New mceliece variants from moderate density parity-check codes. In: 2013 IEEE international symposium on information theory. pp. 2069–2073. IEEE (2013)
44. Monico, C., Rosenthal, J., Shokrollahi, A.: Using low density parity check codes in the mceliece cryptosystem. In: 2000 IEEE International Symposium on Information Theory (Cat. No. 00CH37060). p. 215. IEEE (2000)
45. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Prob. Control and Inf. Theory 15(2), 159–166 (1986)
46. Nojima, R., Imai, H., Kobara, K., Morozov, K.: Semantic security for the mceliece cryptosystem without random oracles. Designs, Codes and Cryptography 49(1-3), 289–305 (2008)
47. Otmani, A., Tillich, J.P., Dallot, L.: Cryptanalysis of two mceliece cryptosystems based on quasi-cyclic codes. Mathematics in Computer Science 3(2), 129–140 (2010)
48. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Annual International Cryptology Conference. pp. 433–444. Springer (1991)
49. Sarwate, D.V.: On the complexity of decoding goppa codes (corresp.). IEEE Transactions on Information Theory 23(4), 515–516 (1977)
50. Sendrier, N., Vasseur, V.: On the decoding failure rate of qc-mdpc bit-flipping decoders. In: International Conference on Post-Quantum Cryptography. pp. 404–416. Springer (2019)
51. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. pp. 124–134. Ieee (1994)
52. Van Keer, R., Viguier, B.: Kangarootwelve: Fast hashing based on keccak-p. In: Applied Cryptography and Network Security: 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings. vol. 10892, p. 400. Springer (2018)
53. Wang, Y.: Rlcekey encapsulation mechanism (rlce-kem) specifcation. NIST Submission (2017)
54. Yamada, A., Eaton, E., Kalach, K., Lafrance, P., Parent, A.: Qc-mdpc kem: A key encapsulation mechanism based on the qc-mdpc mceliece encryption scheme. NIST Submission (2017)